2006

# Modeling Off-the-Shelf Pan/Tilt Cameras for Active Vision Systems

Yuriy Luzanov

# Modeling Off-the-Shelf Pan/Tilt Cameras for Active Vision Systems

by

Yuriy Luzanov

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Associate Professor Dr. Juan C. Cockburn
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
May 2006

**Approved By:**

Juan Cockburn
Dr. Juan C. Cockburn
Associate Professor
Primary Adviser

Marcin Lukowiak
Dr. Marcin Lukowiak
Visiting Assistant Professor, Department of Computer Engineering

Muhammad Shaaban
Dr. Muhammad Shaaban
Associate Professor, Department of Computer Engineering

# Thesis Release Permission Form

Rochester Institute of Technology

Kate Gleason College of Engineering

Title: Modeling Off-the-Shelf Pan/Tilt Cameras for Active Vision Systems

I, Yuriy Luzanov, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or in part.

Yuriy Luzanov

Yuriy Luzanov

Date

# Abstract

There are many existing multicamera systems that perform object identification and tracking. Some applications include but are not limited to security surveillance and smart rooms. Yet there is still much work to be done in improving such systems to achieve a high level of automation while obtaining reasonable performance. Thus far design and implementation of these systems has been done using heuristic methods, primarily due to the complexity of the problem. Most importantly, the performance of these systems is assessed by evaluating subjective quantities. The goal of this work is to take the first step in structured analysis and design of multicamera systems, that is, to introduce a model of a single camera with associated image processing algorithms capable of tracking a target. A single camera model is developed such that it could be easily used as a building block for a multicamera system.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years advances in computing technologies have made it possible for many computer vision applications to run on desktop systems in real time. Also, the cost of the off-the-shelf pan/tilt/zoom cameras has dropped into a reasonable range. Thus, there has been a significant effort in improving existing and developing new computer vision techniques, which could be used for numerous applications from security to human-computer interaction. These techniques have been developed for systems with a single and multiple cameras.

There are numerous advantages of having more than one camera track an object. In the case when an object is occluded from some of the cameras it would still be possible to track it with the remaining ones. This provides some redundancy in the system. In addition, information from multiple cameras can be combined to obtain a 3D location of the object being tracked. With a single camera it is very difficult to determine the location of an object in space. 3D information can be used to compute trajectories, provide accurate path estimation and control action.

Utilizing multiple pan/tilt/zoom cameras to cooperatively track objects in a scene is an active area of research. There has been done a lot of work in building systems to detect and track objects with a single camera, [19] [1] [33]. But the problem of mathematical analysis of multi-camera systems, especially with moving pan/tilt/zoom cameras, has not received much attention.

The goal of this work is to take the first step in the direction of control-theoretic analysis

and design of active multicamera systems. So far a suitable model of a single camera system that combines the dynamical behavior of the pan and tilt actuators and the kinematic aspects of pixels' motion in the image has not been mentioned in the literature. The single camera system model which is developed in this work is the essential building block for multicamera systems.

Furthermore, the camera considered here will be an off-the-shelf pan/tilt camera. It exhibits nonlinear behaviors and provides limited feedback to the user. The cost of such off-the-shelf cameras is considerably less than that of sophisticated robotic arms with linear actuators making it a preferable type of sensor in many applications. In order to control this nonlinear system a robust control framework is used. The goal of the control algorithm is to track a target by keeping it as close as possible to the center of the image. The model and the controller developed in this work are validated by verifying the performance of the closed loop system against the data obtained with an experimental setup.

## 1.1 Previous work

One of the most comprehensive efforts to develop a multi-camera tracking system has been reported in [12]. It is an overview of the results of the Video Surveillance and Monitoring (VSAM) project sponsored by DARPA. The project resulted in a system consisting of a distributed network of surveillance cameras around the Carnegie Mellon University (CMU) campus, intended to provide security personnel only with relevant information about the scene. The automated surveillance system was able to detect regions of interest in the scene by analyzing information from the camera processing units. A virtual 3D environment was created and displayed on a monitor with all the moving objects labelled to make it easier for the user to monitor the scene. A schematic representation of the VSAM system is shown in figure 1.1. OCU is the operator control unit, which receives and integrates the data from remote sensor processing units (SPUs). The user interacts with the system through a Graphical User Interface (GUI). 3-D scene information can also be displayed at remote

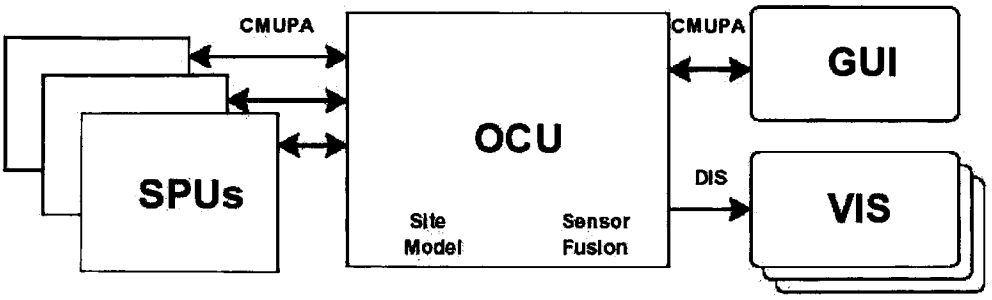



Figure 1.1: Schematic overview of the VSAM system

locations called distributed visualization nodes (VIS). The control of the pan and tilt motion of the cameras was not considered since the pan/tilt platform was extremely slow.

Other work in the area of multi-camera vision systems includes [29],[27],[15]. In all of these projects only high level decision making is emphasized. Control algorithms are developed using heuristics to suit a particular problem. Due to slow time constants of active sensors and processing algorithms a seemingly stable system is achieved. However, there is no formal analysis of the stability or performance of such multicamera systems. The performance aspects are determined experimentally, by observing the behavior of the algorithms implemented in the experimental setup, without considering a mathematical model that incorporates the dynamics of the system's components. This may lead to unreliable systems that may unexpectedly enter unstable or undesired regions of operation since subspaces of possible states are not known.

One control-theoretic, model based analysis and design approach to active vision systems was introduced in [34]. The goal here was to design a system that is insensitive to uncertainties commonly encountered in navigation systems based on computer vision techniques. The result was a controller guaranteeing robust performance under some degree of variation in the parameters (focal length, etc...). The system in this work consisted of a robotic arm with a stereo camera head. A model for the robotic arm and the camera was obtained using a Robust Identification Framework [32]. Note that in contrast to off-the-shelf pan/tilt cameras such systems generally exhibit linear behavior and provide more feedback signals,

thus being easier to model and control at the expense of higher hardware costs.
Another control-theoretic approach to active vision was proposed in [8]. The goal of this work was to track a target with a camera mounted on a robotic manipulator without loosing the points of interest on the target. An image based navigation function was computed resulting in a known imaged based trajectory that achieves the goal of tracking a target without loosing the points of interest. Then an adaptive controller was designed to actually move the camera in a calculated trajectory. Note that here a linear robotic manipulator was used. Also, only the kinematic model of the system was considered for the design of the controller.

## 1.2 Visual tracking system used in the experiments

The goal of this work is not to introduce novel computer vision techniques, but to focus on mathematical analysis and design of a class of active vision systems which perform object detection and tracking. As a consequence, the experimental system used here is a simple target detection and tracking application representative of such class of systems.
As mentioned earlier, the system under consideration incorporates an off-the-shelf pan/tilt camera. To be more specific, a SONY EVI-D100 pan/tilt/zoom camera is used. The target to be tracked is a red ball in a laboratory (indoors) environment. The general idea of the detection algorithm is to find the largest red object in the image. The center of the bounding box around the object defines its location in the image plane. If there is not a red object in the image that satisfies the geometric constraints of the ball, then the current location of the object is set to the center of the image, so that the camera stops moving. After obtaining an image from the video stream it takes 55 ms for the image processing algorithms to complete, yielding a minimum sampling time of 55 ms. The amount of time it takes to process an image is related to its content and to its size. Here it will be assumed that we know the centroid of the target in the image, thus the details of image processing techniques are not the concern of this work.

## 1.3 Overview

This document is organized in the following way. In the second chapter, image processing algorithms are modeled using optical flow techniques. Then, dynamic properties of the pan/tilt camera are incorporated into the model and an uncertain model of the overall system is developed. In the third chapter this model is expanded for the design of a robust controller to cope with many uncertainties present in off the shelf pan/tilt cameras. The simulations of the model and the controller are verified with the experimental results. In chapter four an overview of necessary steps to expand the single camera model into a multicamera one is presented. Finally, chapter five summarizes the contributions of the work, and gives possible directions for future work.

# Chapter 2

# Modeling a Single Camera System

This chapter addresses different issues in modeling a single off-the-shelf camera system. First, a pixel motion model that describes the geometry of the image formation process is presented. Then, state-space equations for the pan and tilt dynamics of the camera are derived. Finally, the pan and tilt dynamics are combined with the pixel motion model to form a nonlinear model of the camera.

## 2.1 Pixel Motion Model

A pixel motion model relates the motion of a pixel in the image to the motion of a physical object in a 3D scene and to the motion of the camera. A perspective projection model with a pin-hole camera was assumed for the derivation of the equations. This is a simple model and it gives a sufficient approximation for the geometry of image formation without optical distortions. The camera is assumed to be fixed. Therefore, it can only pan and tilt and does not have a translational velocity. The object being tracked can be described by a bounding box. Since zoom is assumed to be constant just the center of the bounding box is sufficient to identify the position of the object in the image for the purposes of tracking. Thus, the object is modeled as a point in space and does not have a rotational velocity associated with it.

The projection of the object in the image will move due to the motion of the object and also due to the motion of the camera. Thus, to properly model the motion of the projection of the

object in the image both, the translational velocity of the object and the rotational velocity of the camera will have to be considered. The following are the perspective projection relationships as illustrated in figure 2.1 and the motion of a point in an image due to translation and rotation in 3D space, [30] [31]:

$$x = \frac{fX_s}{Z_s\gamma_x}; \quad y = \frac{fY_s}{Z_s\gamma_y} \tag{2.1}$$

$$\frac{d\mathbf{P}}{dt} = \mathbf{T} - \mathbf{R} \times \mathbf{P} \tag{2.2}$$

where $f$ is the focal length, measured in meters; $\gamma$ is the physical length of the pixel, measured in meters; $\mathbf{P} = (X_s, Y_s, Z_s)^T$ is the location of the point in 3D space, measured in meters; $x$ and $y$ are the coordinates of the point's projection in the image plane, measured in pixels; $\mathbf{T} = (T_x, T_y, T_z)^T$ is the vector of translational velocities of the point, measured in meters per second; $\mathbf{R} = (R_x, R_y, R_z)^T$ is the vector of rotational velocities of the camera, measured in radians per second; $f$ and $\gamma$ are the intrinsic camera parameters.

The cross product $\mathbf{R} \times \mathbf{P}$ is equal to:

$$\mathbf{R} \times \mathbf{P} = \begin{bmatrix} 0 & -R_z & R_y \\ R_z & 0 & -R_x \\ -R_y & R_x & 0 \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} \tag{2.3}$$

Using these relationships the motion of the projection of a point in the image, $\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$, due to the rotational motion of the camera and the translational motion of the point can be found as follows:

$$\frac{dx}{dt} = \frac{f}{\gamma_x}\frac{d(\frac{X_s}{Z_s})}{dt} = \frac{f}{\gamma_x}\frac{Z_s\frac{dX_s}{dt} - X_s\frac{dZ_s}{dt}}{Z_s^2} \tag{2.4}$$

$$\frac{dX_s}{dt} = T_x - R_yZ_s + R_zY_s \tag{2.5}$$

$$\frac{dZ_s}{dt} = T_z - R_xY_s + R_yX_s \tag{2.6}$$

$$\begin{aligned} \frac{dx}{dt} &= \frac{f}{\gamma_x}\frac{Z_s(T_x - R_yZ_s + R_zY_s) - X_s(T_z - R_xY_s + R_yX_s)}{Z_s^2} \\ &= \frac{fT_x}{Z_s\gamma_x} - x\frac{T_z}{Z_s} + xy\frac{\gamma_y}{f}R_x - (\frac{f}{\gamma_x} + x^2\frac{\gamma_x}{f})R_y + \frac{\gamma_y}{\gamma_x}yR_z \end{aligned} \tag{2.7}$$

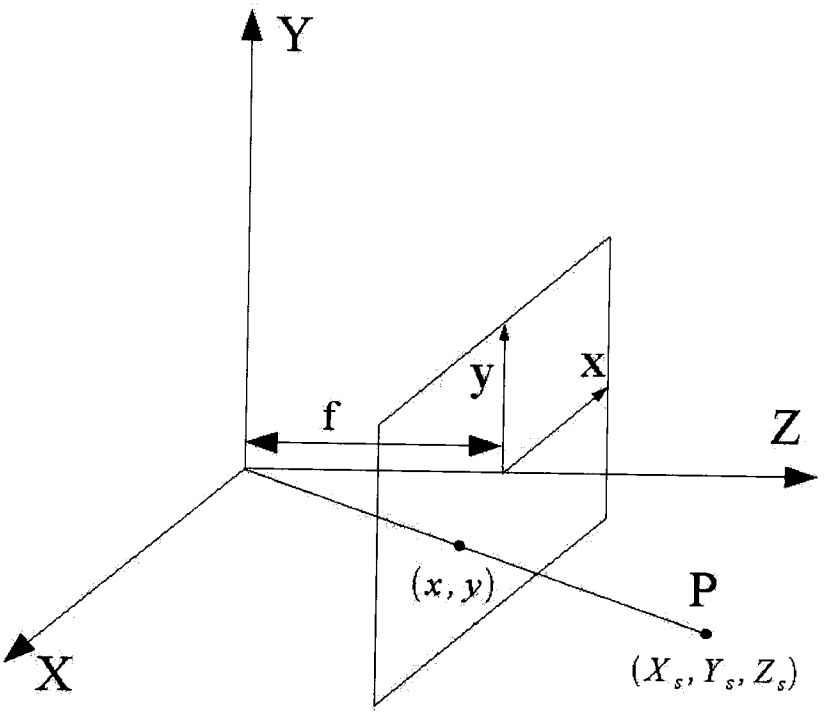



Figure 2.1: Perspective Projection Diagram

$$\frac{dY_s}{dt} = T_y + R_x Z_s - R_z X_s \tag{2.8}$$

$$\begin{aligned}\frac{dy}{dt} &= \frac{f}{\gamma_y}\frac{Z_s\frac{dY_s}{dt} - Y_s\frac{dZ_s}{dt}}{Z_s^2} \\ &= \frac{f}{\gamma_y}\frac{Z_s(T_y + R_x Z_s - R_z X_s) - Y_s(T_z - R_x Y_s + R_y X_s)}{Z_s^2} \\ &= \frac{fT_y}{Z_s\gamma_y} - y\frac{T_z}{Z_s} + (\frac{f}{\gamma_y} + y^2\frac{\gamma_y}{f})R_x - xy\frac{\gamma_x}{f}R_y - x\frac{\gamma_x}{\gamma_y}R_z \end{aligned} \tag{2.9}$$

Using the results from 2.7 and 2.9 the pixel velocities can be written in matrix form as:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\frac{f}{Z_s\gamma_x} & 0 & -\frac{x}{Z_s}\\ 0 & \frac{f}{Z_s\gamma_y} & -\frac{y}{Z_s}\end{bmatrix}\begin{bmatrix}T_x\\ T_y\\ T_z\end{bmatrix} + \begin{bmatrix}xy\frac{\gamma_y}{f} & -(\frac{f}{\gamma_x} + x^2\frac{\gamma_x}{f}) & y\frac{\gamma_y}{\gamma_x}\\ \frac{f}{\gamma_y} + y^2\frac{\gamma_y}{f} & -xy\frac{\gamma_x}{f} & -x\frac{\gamma_y}{\gamma_x}\end{bmatrix}\begin{bmatrix}R_x\\ R_y\\ R_z\end{bmatrix} \tag{2.10}$$

where we have decoupled the effects of translational and rotational motion.

This model describes the kinematic aspects of the motion of the pixel in the image plane. It will later be integrated with the dynamic properties of the camera platform to obtain a complete representation of the vision system.

## 2.2 Modeling the Dynamics of the Camera

In the pixel motion model presented above, the camera is assumed to have no dynamics. Only the kinematic relationships, such as the rotational velocity of the camera, were considered. This implies that a physical camera unit can instantaneously follow the reference input. Such an assumption is not realistic, since objects that are tracked may move quickly relative to the time constants of the camera. In order to build a system that fully utilizes the capabilities of the camera, a dynamic model must be considered.

### 2.2.1 Identifying the Transfer Function of the System

In this section the camera dynamics will be approximated as a linear time invariant (LTI) system with delay.

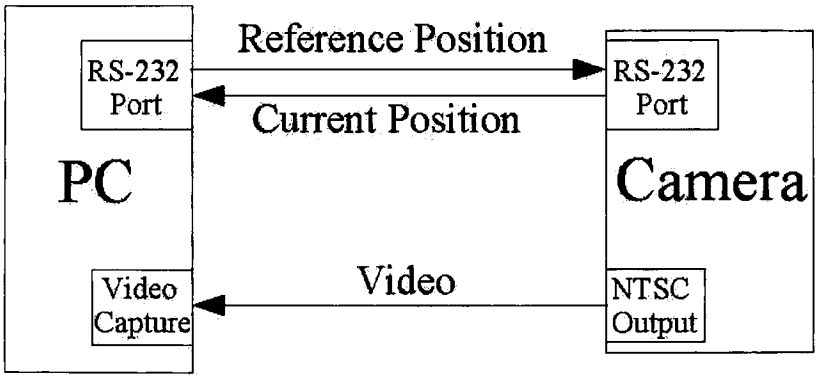



Figure 2.2: Sony EVI-D100 Camera Interconnection with the Processing Unit

Assuming that the camera is a linear system, the dynamic behavior from the reference position input to the current position output, as seen in Figure 2.2, can be described by a general transfer function model:

$$H(s) = \frac{n(s)}{d(s)} \tag{2.11}$$

First, a sufficient number of poles and zeros required to represent the dynamics of the camera need to be identified. This corresponds to the degree of the numerator, $n$, and degree of the denominator, $d$.

Consider a general off-the-shelf pan/tilt camera, Figure 2.2: the pan and tilt motions are controlled by supplying a reference position input, which is the desired angular position for the pan and tilt angles, and the only available outputs related to the dynamics of the camera are the current pan and tilt angles. Once the reference input is supplied, the camera moves to the specified position with a zero steady state tracking error. This suggests that the pan/tilt system must have an internal feedback loop with integral action to achieve zero steady state error.

Figure 2.3 illustrates the step response obtained from panning the Sony EVI-D100 camera 50 degrees. The system has zero steady state tracking error, as mentioned earlier. Note also, that there is a short delay of about 25 msec. It consists of the communication delay over the RS-232 link, which was measured to be on average 15 msec, and the processing delay internal to the camera platform.

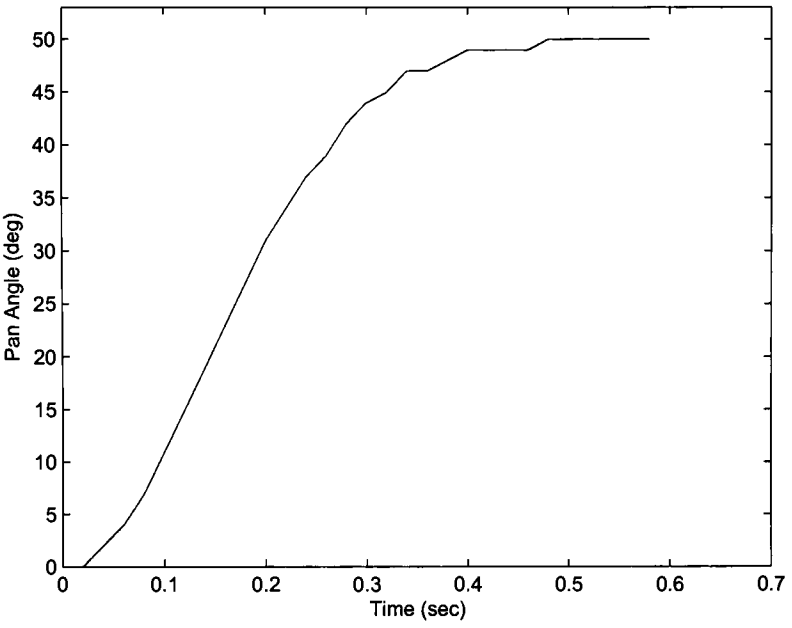



Figure 2.3: Step response of Sony EVI-D100, $T = 0.02\ sec$

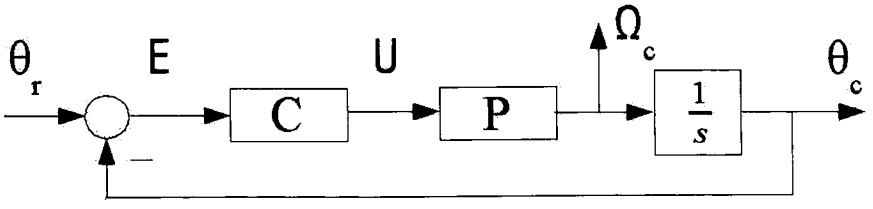



Figure 2.4: Sony model block diagram

The block diagram in Figure 2.4 is a simplified representation of the closed loop position control system internal to the camera. Only the reference input, $\Theta_r$, and the output signal, $\Theta_c$, are accessible from the outside world. Dynamics associated with the motor and the camera's mass are represented by block $P$. The camera platform consists of a mass mounted on an electrical motor, this type of assembly is usually dominated by a slow mechanical pole. Therefore, the camera's dynamics can be approximated by a first order system:

$$P(s) = \frac{K_p}{\tau s + 1} \tag{2.12}$$

The input into $P(s)$ is the control action, $U(s)$, generated by the controller, $C(s)$. The output of $P(s)$ is the angular velocity of the camera, $\Omega_c(s)$. It is integrated to obtain the angular position $\Theta_c(s)$, which is subtracted from the reference input to obtain the error signal, $E(s)$. The error signal is the input to the controller. The open loop is stable and has an integrator thus a proportional controller is sufficient for an approximate model. Let the controller gain be $C(s) = K_c$, where $K_c$ includes the gain of the plant, $K_p$. Then, the closed loop transfer function from the reference input, $\Theta_r(s)$, to the camera output, $\Theta_c(s)$, is:

$$\frac{\Theta_c(s)}{\Theta_r(s)} = \frac{\frac{1}{\tau s+1}\frac{1}{s}K_c}{1+\frac{1}{\tau s+1}\frac{1}{s}K_c} \tag{2.13}$$

$$= \frac{K_c}{s(\tau s+1)+K_c} \tag{2.14}$$

$$= \frac{K_c}{\tau s^2+s+K_c} \tag{2.15}$$

$$= \frac{K}{(s+p_1)(s+p_2)} \tag{2.16}$$

where,

$$K = \frac{K_c}{\tau} \tag{2.17}$$

$$p_1 = \frac{\sqrt{1-4\tau K_c}-1}{2\tau} \tag{2.18}$$

$$p_2 = \frac{-\sqrt{1-4\tau K_c}-1}{2\tau} \tag{2.19}$$

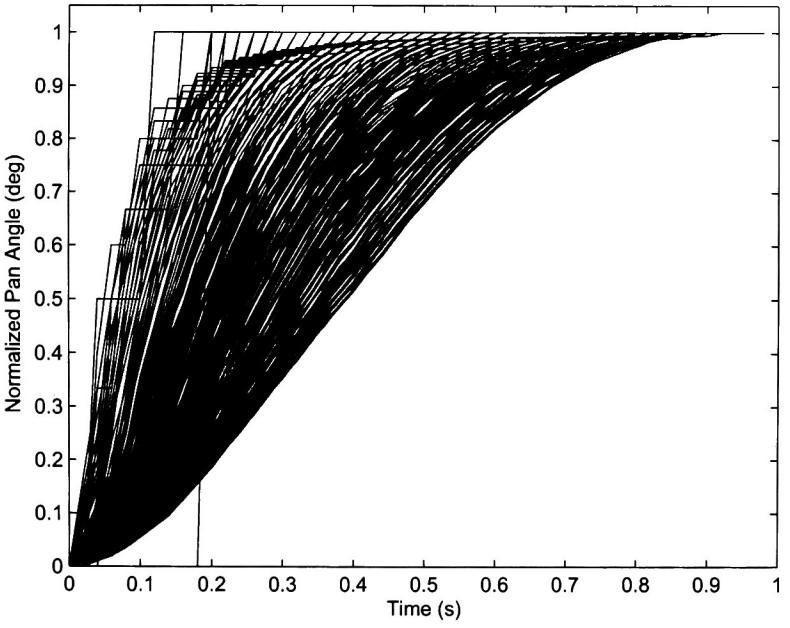



Figure 2.5: Response of SONY EVI-D100 to a range of steps from $1^o$ to $200^o$

In equation 2.16, $p_1$ and $p_2$ are the only two poles required to represent the system without taking into account any delay. Both poles have to be stable and real, as can be seen from the step response of the camera in Figure 2.3. The overall gain of the system is $K$.

Finally, the delay can be modeled using a first order *Padé* approximation resulting in the final transfer function:

$$\frac{\Theta_c(s)}{\Theta_r(s)} = \frac{-s+z}{s+p} \frac{K}{(s+p_1)(s+p_2)} \tag{2.20}$$

Thus, to represent the dynamics of the camera under consideration, it is sufficient to consider a class of transfer functions with the order of the numerator $n = 1$ and the order of the denominator $d = 3$.

### 2.2.2 SONY EVI-D100 dynamic camera model

Now, that the class of transfer functions under consideration is identified, it is time to look at the possible values for the model parameters: $z$, $p$, $p_1$, $p_2$ and $K$. These values differ depending on the size of the step in the step response of the camera as shown in Figure 2.5. The variation in the parameters is a result of a nonlinear behavior of the system. Also, with small step sizes, less than 5 degrees, the response tends to be dominated by a delay, resulting in the outliers in Figure 2.5. As was discussed earlier, the delay and both poles' locations vary with the step size. Thus, $z$, $p$, $p_1$, $p_2$ and $K$ will not be constants, but will be defined on an interval. To find the values of the parameters from a particular step response a simple technique described in [16] is used. It was implemented as a Matlab function and included in Appendix E.2. The intervals for the parameters are determined by the slowest and fastest step responses and are:

$$\begin{aligned} z &\in [20, 2000] \\ p &\in [20, 2000] \\ p_1 &\in [9.6, 13.2] \\ p_2 &\in [9.8, 50.6] \\ K &\in [94.6, 668.4] \end{aligned}$$

Let the nominal plant correspond to a step response of a $25^o$ step. The observed delay with $25^o$ step is 0.01 sec. Using this information the nominal plant transfer function is found to be:

$$\frac{\Theta_c}{\Theta_r} = \frac{-s + 200}{s + 200} \frac{311.8}{(s + 13.6)(s + 22.9)} \tag{2.21}$$

It is worth noting that the values of $z$ and $p$ will always be equal and $z$ will be located in the right half plane leading to a non minimum phase system. The response to a step of $25^o$ of the above transfer function and the experimental step response data are compared in figure 2.6. It is clear from the figure that a two pole approximation with delay is sufficient to represent the physical system.

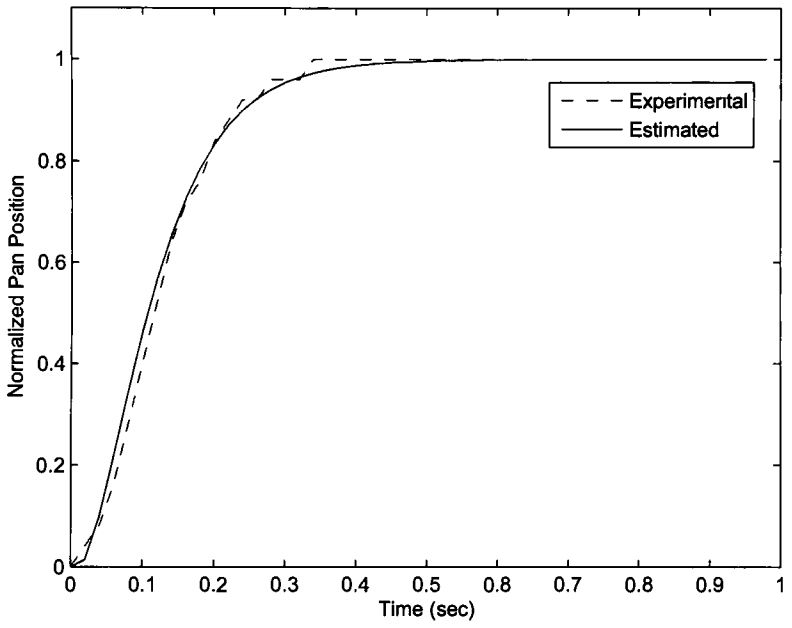



Figure 2.6: Experimental and Estimated Response to a 25° Step

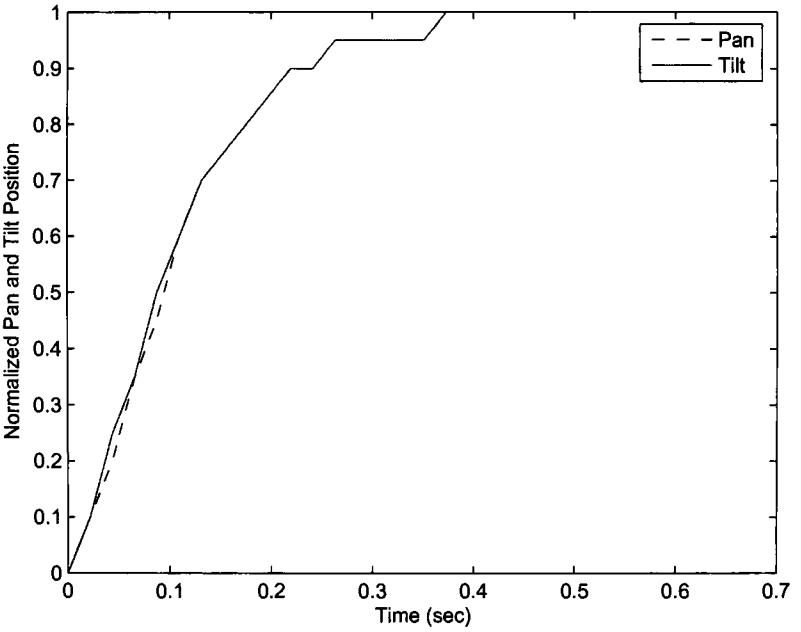



Figure 2.7: Comparison of Normalized $20^o$ Pan and Tilt Steps

In equations 2.20 and 2.21 only the pan of the camera is considered. The tilt dynamics are very similar to the pan dynamics as seen in Figure 2.7, where a pan of $20^o$ is compared to the tilt of $20^o$. Thus, all equations derived here and in the future for the pan dynamics are also valid for the tilt dynamics.

## 2.3 Integration of the Dynamic Camera Model with the Pixel Motion Model

In order to integrate the two models derived in the previous sections it is necessary to represent the dynamic camera model in equation 2.20 in a state-space form. In equation 2.20, there are two poles associated with the two time constants and a pole/zero combination as a result of the *Padé* approximation, creating a total of three states. The output of the

equation is the angular position of the camera $\Theta_c$. The pixel motion model, 2.10, requires an angular velocity of the camera as one of the parameters. Thus, another output $\Omega_c$, the angular velocity of the camera, needs to be added in the state-space model. The following is the resulting state-space model of the dynamics of the camera:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Theta_r \tag{2.22}$$

$$\begin{bmatrix} \Theta_c \\ \Omega_c \end{bmatrix} = \begin{bmatrix} 0 & c_{12} & c_{13} \\ c_{21} & c_{22} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.23}$$

where,

$$\begin{aligned} a_{11} &= -(p + p_1 + p_2) \\ a_{12} &= -(pp_1 + pp_2 + p_1p_2) \\ a_{13} &= -pp_1p_2 \\ c_{12} &= c_{21} = -K \\ c_{13} &= c_{22} = Kz \end{aligned}$$

from equation 2.20.

The pan equation will be denoted by subscript $x$. In a compressed form it will be:

$$\dot{\mathbf{x}}_\mathbf{x} = A_x\mathbf{x}_\mathbf{x} + B_x\Theta_{rx} \tag{2.24}$$

$$\mathbf{y}_\mathbf{x} = C_x\mathbf{x}_\mathbf{x} \tag{2.25}$$

Since the tilt motion has similar dynamics to the pan motion, the tilt equation will be the same as pan equation with slightly different $A$, $B$ and $C$ matrices. Symbolically, the state-space equation for tilt will be denoted with subscript $y$ and in a compressed form will be:

$$\dot{\mathbf{x}}_\mathbf{y} = A_y\mathbf{x}_\mathbf{y} + B_y\Theta_{ry} \tag{2.26}$$

$$\mathbf{y}_\mathbf{y} = C_y\mathbf{x}_\mathbf{y} \tag{2.27}$$

Combining equation 2.10 with the dynamical camera equations for pan and tilt results in the following state-dependent state-space equations that have both the dynamics and kinematics of the camera system:

$$\dot{\mathbf{z}} = A(\mathbf{z})\mathbf{z} + B\mathbf{u} + E(\mathbf{z})\mathbf{v} \tag{2.28}$$

$$\mathbf{y} = C\mathbf{z} + \eta \tag{2.29}$$

where the measurement $\mathbf{y}$ is corrupted by noise $\eta$.

In 2.28 and 2.29 $\mathbf{z}$ is the vector of states:

$$\mathbf{z} = \begin{bmatrix} x & y & p_{1x} & p_{2x} & p_{3x} & p_{1y} & p_{2y} & p_{3y} \end{bmatrix}^T \tag{2.30}$$

The input vector $\mathbf{u}$ is composed by the reference pan/tilt positions

$$\mathbf{u} = \begin{bmatrix} \Theta_{rx} & \Theta_{ry} \end{bmatrix}^T \tag{2.31}$$

The translational velocity of the target is modeled as a disturbance and is given by

$$\mathbf{v} = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T \tag{2.32}$$

The vector of outputs $\mathbf{y}$ is given by:

$$\mathbf{y} = \begin{bmatrix} x & y & \Theta_x & \Theta_y \end{bmatrix}^T \tag{2.33}$$

The state matrix $A(\mathbf{z})$ is:

$$\begin{bmatrix}
0 & 0 & c_{21x}xyn_y & c_{22x}xyn_y & 0 & -c_{21y}(\frac{1}{n_x} + x^2 n_x) & -c_{22y}(\frac{1}{n_x} + x^2 n_x) & 0 \\
0 & 0 & c_{21x}(\frac{1}{n_y} + y^2 n_y) & c_{22x}(\frac{1}{n_y} + y^2 n_y) & 0 & -c_{21y}xyn_x & -c_{22y}xyn_x & 0 \\
0 & 0 & a_{11x} & a_{12x} & a_{13x} & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & a_{11y} & a_{12y} & a_{13y} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix} \tag{2.34}$$

where $n_x = \frac{\gamma_x}{f}$ and $n_y = \frac{\gamma_y}{f}$ are intrinsic camera parameters; $c_{ijx}$ and $c_{ijy}$ are elements of the uncertain output matrices, $C_x$ and $C_y$; $a_{ijx}$ and $a_{ijy}$ are elements of the uncertain state matrices, $A_x$ and $A_y$; $x$ and $y$ are the states. Thus, $A$ is a nonlinear uncertain state-dependent matrix.

The input matrix $B$ is:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{2.35}$$

The disturbance matrix $E(\mathbf{z})$ is:

$$\begin{bmatrix} \frac{1}{Z_s n_x} & 0 & -\frac{x}{Z_s} \\ 0 & \frac{1}{Z_s n_y} & -\frac{y}{Z_s} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.36}$$

where $x$ and $y$ are the states. The distance to the object, $Z_s$, will be considered an uncertain parameter. Thus, $E$ is a nonlinear uncertain state-dependent disturbance matrix.

Finally, the output matrix $C$ is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{12x} & c_{13x} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{12y} & c_{13y} \end{bmatrix} \tag{2.37}$$

where $c_{ijx}$ and $c_{ijy}$ are elements of uncertain matrices $C_x$ and $C_y$.

In summary, the model of the camera system described above is a nonlinear uncertain model with state-dependent state-space matrices. The inputs into the model are the desired pan and tilt positions of the camera. The outputs are the current pan and tilt positions of the

camera and the position of the target in the image plane in pixels. The object's translational velocity, $T$, enters the model as an uncontrollable but bounded disturbance. The output is corrupted by noise, $\eta$, which enters the system primarily through the image processing algorithms used to compute the location of the target in the image. A summary of the state-space equations is given in Appendix A.

# Chapter 3

# Robust Control of a Single Camera Tracking System

The behavior that the final system has to exhibit is the tracking of a traget. The model of the pan/tilt camera derived in the previous section is an uncertain model with parametric uncertainty. A control systems framework developed to handle uncertain models is called robust control. Thus, the robust control framework will be used to design a tracking controller for the uncertain model of the camera.

## 3.1 Overview of Robust Control Framework

All physical systems exhibit uncertain behaviors to some extend. One of the major sources of uncertainty is imprecise component values inherent to the manufacturing processes. In many instances approximation of the physical systems with deterministic models without consideration of uncertainties is sufficient for the task a hand. But in high performance systems small changes in parameters can cause significant degradation of performance. Models of such systems have to incorporate the information about the parameter variations to be useful for purposes of control. The robust control framework offers a way to model both the unknown and the known behaviors of the system. The unknown behavior is modeled as the uncertainty.

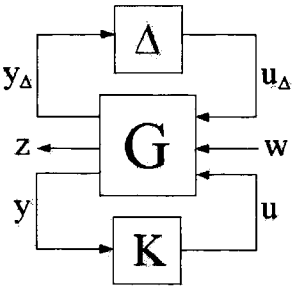



Figure 3.1: Uncertainty, Controller and Generalized Plant Interconnection Diagram

The structure of the uncertainty is represented by the $\Delta$ block in figure 3.1. In the same figure, $G$ is the generalized plant and $K$ is the controller. The generalized plant incorporates the system dynamics, the objectives to be achieved by the controller and some information about the uncertainties. Signal $w$ is the vector of exogenous inputs into the system. The output $z$ is the vector of error signals to be minimized. In general, the robust control problem is to find a stabilizing controller, $K$, such that some $p$-norm of the transfer function from $w$ to $z$, $T_{zw}$, is minimized. This can be written as:

$$\min_{K} ||T_{zw}||_p \tag{3.1}$$

The details on how to obtain the uncertainty matrix, $\Delta$, and the generalized plant, $G$, for the problem at hand are given in the following sections. A good reference for robust control theory can be found in [38].

## 3.2 Derivation of the uncertain model

In order to be able to study the system with parametric uncertainty derived in the earlier section, it has to be converted into a Linear Fractional Representation (LFR) [11] [10]. In a linear fractional form, the model has a constant martrix, $M$, interconnected with a structured uncertainty matrix, $\Delta$, as illustrated in Figure 3.2. There are lower LFRs and upper LFRs, which differ in the way the feedback enters the $M$ matrix. This is illustrated in Figure 3.2, where an upper LFR is depicted on the left and a lower LFR is depicted on

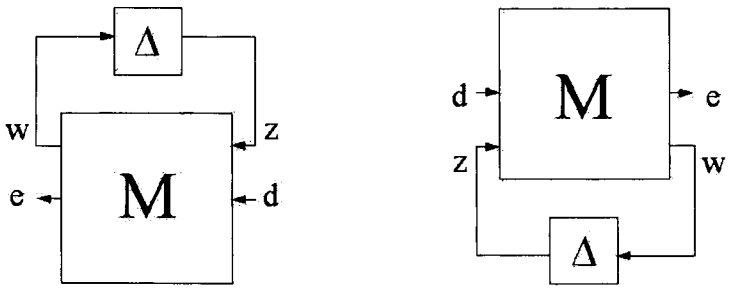



Figure 3.2: Upper (on the left) and Lower (on the right) Linear Fractional Representations (LFRs)

the right. An upper or a lower linear fractional transformation (LFT) is used to obtain a LFR. The following are the definitions of $\mathcal{F}_u$, the upper LFT, and $\mathcal{F}_l$, the lower LFT,

$$\mathcal{F}_u(M, \Delta) = M_{22} + M_{21}\Delta(I - M_{11}\Delta)^{-1}M_{12} \tag{3.2}$$

$$\mathcal{F}_l(M, \Delta) = M_{11} + M_{12}\Delta(I - M_{22}\Delta)^{-1}M_{21} \tag{3.3}$$

Here, an upper LFR will be considered for the uncertain model. Let $\boldsymbol{q}$ represent a vector of uncertain parameters. Then, the state space equations from 2.28 and 2.29 can be rewritten in the following form:

$$\begin{bmatrix} \mathbf{y} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} D(q) & C(q) \\ B(q) & A(q) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{z} \end{bmatrix} \tag{3.4}$$

Note that $B(q)$ is now the combination of $B$ and $E(z)$ in equation 2.28. Now,

$$\Sigma(\boldsymbol{q}) = \begin{bmatrix} D(q) & C(q) \\ B(q) & A(q) \end{bmatrix} \tag{3.5}$$

where, $\boldsymbol{q}$ is the vector of uncertain parameters.

Finding an LFR of the system is equivalent to finding a $(M, \Delta)$ pair, such that:

$$\Sigma(q) = M_{22} + M_{21}\Delta(q)(I - M_{11}\Delta(q))^{-1}M_{12} \tag{3.6}$$

where $\Delta(q)$ is a diagonal matrix of uncertainties. In general this can be expressed as:

$$\Sigma(q) = \Delta(q) \star M \tag{3.7}$$

where $\star$ represents the Redheffer star product. For two systems, $N$ and $M$, the Redheffer star product is given by:

$$N \star M = \begin{bmatrix} \mathcal{F}_l(N, M_{11}) & N_{12}(I - M_{11}N_{22})^{-1}M_{12} \\ M_{21}(I - N_{22}M_{11})^{-1}N_{21} & \mathcal{F}_u(M, N_{22}) \end{bmatrix} \tag{3.8}$$

One technique used to find LFRs from state space equations is the Structured Tree Decomposition [9]. The result of applying this procedure to 3.5 are matrices $M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$ and $\Delta(q)$, which are attached in Appendix B. The values of the parameters in the above matrices are summarized in Table 3.1.

### 3.2.1 Normalization of the variables and model reduction

In order to make the structured tree decomposition process simpler, the uncertain parameters were mapped to $q_i$ and constants to $k_i$. The model will be linearized around the equilibrium by letting $x$ and $y$ be uncertain on a small interval around the equilibrium. The purpose of the tracking algorithm is to keep the target at the center of the image, which is $(0, 0)$ in pixel coordinates. Thus, $(0, 0)$ will be the center of the uncertain interval. The intervals are represented in the form $(a_i, b_i)$, where $a_i$ is the center and $b_i$ is the radius for the $i$'s parameter. With such a representation of the intervals, $q_i$ can be expressed as:

$$q_i = a_i + b_i\delta_i, \quad |\delta_i| \leq 1 \tag{3.9}$$

The mappings of the uncertain parameters and the constants, with the intervals of the uncertain parameters and values of constants are summarized in Table 3.1.

Following the procedure for the normalization of the parameters outlined in [11], $\Sigma(q)$ can be expressed as:

$$\Sigma(q) = \Delta(\delta) \star \tilde{L} \tag{3.10}$$

where $\Delta(\delta)$ is matrix of normalized uncertainties $\delta_i$ and $\tilde{L}$ is the corresponding matrix that incorporates the information about the uncertainties' ranges and intervals. Matrix $\tilde{L}$ can be

| Parameter Name | Mapping | Interval/Value | Units |
|---|---|---|---|
| $n_x$ | $k_1$ | 0.005 | m/m |
| $n_y$ | $k_2$ | 0.005 | m/m |
| $\frac{1}{n_x}$ | $k_3$ | 200 | m/m |
| $\frac{1}{n_y}$ | $k_4$ | 200 | m/m |
| conversion constant | $k_5$ | $\pi/180$ | rad/deg |
| x | $q_1$ | (0, 5) | pixels |
| y | $q_2$ | (0, 5) | pixels |
| $a_{11x}$ | $q_3$ | (-68.9, 6.89) | - |
| $a_{12x}$ | $q_4$ | (-1363, 136.3) | - |
| $a_{13x}$ | $q_5$ | (-8291, 829.1) | - |
| $a_{11y}$ | $q_6$ | (-68.9, 6.89) | - |
| $a_{12y}$ | $q_7$ | (-1363, 136.3) | - |
| $a_{13y}$ | $q_8$ | (-8291, 829.1) | - |
| $c_{21x}$ | $q_9$ | (-207.3, 20.73) | - |
| $c_{22x}$ | $q_{10}$ | (8291, 829.1) | - |
| $c_{21y}$ | $q_{11}$ | (-207.3, 20.73) | - |
| $c_{22y}$ | $q_{12}$ | (8291, 829.1) | - |
| $c_{12x}$ | $q_{13}$ | (-207.3, 20.73) | - |
| $c_{13x}$ | $q_{14}$ | (8291, 829.1) | - |
| $c_{12y}$ | $q_{15}$ | (-207.3, 20.73) | - |
| $c_{13y}$ | $q_{16}$ | (8291, 829.1) | - |
| $\frac{1}{Z_s}$ | $q_{17}$ | (0.5, 0.2) | 1/m |

Table 3.1: Constants and Uncertain Parameters

obtained as follows:

$$\tilde{L} = N \star M \tag{3.11}$$

$$N = \left[\begin{array}{c|c} N_{11} & N_{12} \\ \hline N_{21} & N_{22} \end{array}\right] = \left[\begin{array}{c|c} O_r & I_r \\ \hline \oplus_{i=1}^{j} b_i I_{r_i} & \oplus_{i=1}^{j} a_i I_{r_i} \end{array}\right] \tag{3.12}$$

In equation 3.12 $N_{21}$ and $N_{22}$ encode the uncertain parameter ranges given in Table 3.1.

The system, $\tilde{L}$, obtained from equation 3.11 does not have the lowest order due to the fact that the Structured Tree Decomposition does not result in a minimal representation of the uncertain system. Usually the order of the uncertain model can be reduced using special model reduction techniques.

In this case the LFR model was reduced from 32 to 21 uncertain parameters using a generalization of the Kalman decomposition to n-D systems followed by truncation [13]. This method removes "uncontrollable and unobservable states", resulting in an equivalent system of a lesser degree. A function implementing this technique can be found in the *LFR toolbox for use with Matlab* [26]. It is called *minlfr*.

In order to use this function LFR $\tilde{L}$ has to be expressed as an *LFR Toolbox* object.

There is a slight difference in the notation used in this work and in the LFR toolbox. In equation 3.4, the state matrix $A(q)$ is placed in the lower right, so that in the LFR obtained from equation 3.11 the states are in the lower right portion of the matrix. But, the LFR toolbox expects the states in the upper left portion of the system matrix. This mismatch in notation is easily fixed by shifting the appropriate columns and rows of the matrix $\tilde{L}$ as illustrated below:

$$\tilde{L} = \begin{bmatrix} D_{22} & D_{21} & C_2 \\ D_{12} & D_{11} & C_1 \\ B_2 & B_1 & A \end{bmatrix} \Rightarrow L = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}$$

The resulting LFR, $L$, is organized in the following way:

$$L = \left[\begin{array}{c|c} L_{11} & L_{12} \\ \hline L_{21} & L_{22} \end{array}\right] = \left[\begin{array}{cc|c} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ \hline C_2 & D_{21} & D_{22} \end{array}\right] \tag{3.13}$$

where $A$ is the state matrix, $B_1$ is the uncertainty input matrix and $B_2$ is the model input matrix.

One more bit of information is needed to represent the system using the LFR toolbox. When the parameters are normalized, the uncertainties are mapped to a unit ball. In order to represent normalized uncertainties only their positions in the $\Delta(q)$ matrix are required. In the LFR toolbox, this is accomplished with a block matrix. The following block matrix describes all of the uncertain parameters in $\Delta$:

$$blk = \begin{bmatrix} -8 & 0 \\ -3 & 0 \\ -3 & 0 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ -3 & 0 \\ -3 & 0 \\ -3 & 0 \\ -3 & 0 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ -1 & 0 \\ -4 & 0 \end{bmatrix} \tag{3.14}$$

The first row of a block matrix gives the number of states in the system, the size of the state matrix, $A$. Here there are 8 states, thus, the first row is $[-8\ 0]$. The following rows describe the uncertain parameters. If, in the first column there is a negative number, it means that the uncertainty is real. If it is positive, then the uncertainty is complex. For example, in

the second row of the block matrix the entry is $[-3\ 0]$, which means that the first uncertain parameter is real and is repeated 3 times in the $\Delta$ matrix. If the entry in the second column of a block matrix is not zero, then the uncertainty is full block. Here, the uncertainties are all diagonal matrices, hence all the values in the second column are set to zero.
An *lfr* object is obtained using the following LFR toolbox command:

```
sys = lfr(A,B,C,D,blk)
```

where

$$A,\ B = \begin{bmatrix} B_1 & B_2 \end{bmatrix},\ C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix},\ D = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$$

are from equation 3.13.
Now, the model reduction is performed using the following matlab code:

```
sys_red = minlfr(sys)
```

The values of $L_{red}$ and $\Delta_{red}$, which are contained in *sys_red*, are attached in Appendix C. Matrix $L_{red}$ is organized in the same manner as 3.13, that is:

$$L_{red} = \left[\begin{array}{c|c} L_{11} & L_{12} \\ \hline L_{21} & L_{22} \end{array}\right] = \left[\begin{array}{cc|c} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ \hline C_2 & D_{21} & D_{22} \end{array}\right] \tag{3.15}$$

### 3.2.2 Model Verification

In order to ensure that the behavior of the model is close to that of the physical system, the following experiment was conducted: while keeping the target fixed in the viewing range of the camera, the camera was given a command to pan $50^o$ and the position of the target in the image was recorded at each sampling time. This is a step response from the angular position input of the system to the object's coordinate in the image. It verifies both parts of the model — the pixel motion and the pan dynamics.

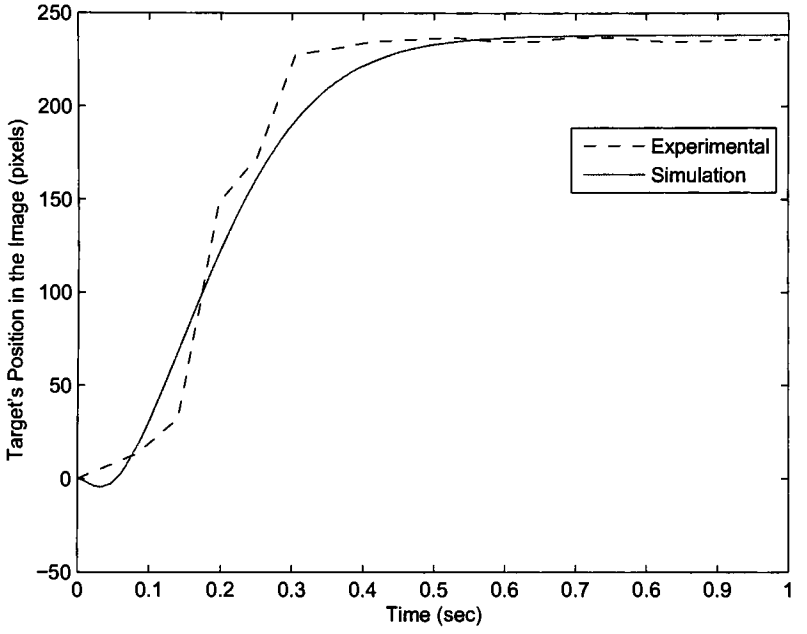



Figure 3.3: Comparison of the Experimental and Simulation Step Responses

The results of the experiment and simulation with the nominal plant are presented in Figure 3.3. The nominal plant is extracted from 3.15, and is:

$$Plant\_0 = \begin{bmatrix} A & B_2 \\ C_2 & D_{22} \end{bmatrix} \tag{3.16}$$

Note, that the nominal plant is continuous, while the experimental system is discrete, with sampling time being equal to the time it takes to process one image in the video stream. In this case the sampling time is approximately $55\ ms$. While there are some differences in the transient responses, the difference in the steady state behaviors is insignificant. The transient response error is contributed primarily by the *Padé* approximation of the delay. The error of about 1 pixel in the steady state response comes from the noise in the images. Overall, the nominal plant gives a good approximation of the physical system.

## 3.3 Closed Loop Controller Design for Robust Tracking

The LFR model developed in the previous section will now be used to synthesize a robust controller to keep the target centered in the image by paning and tilting the camera.

There are total of five inputs into the LFR model. Two of them are the control inputs to the camera system, the reference positions for the pan and tilt actuators. The other three are the disturbance inputs and are the translational velocities of a target with respect to the camera's axes. Also, there are four outputs available from the LFR model. Two of the outputs are the positions of the target in the image plane, one per each dimension. And the other two outputs are the pan and tilt angles of the camera. Thus, the controller will be a Multi-Input-Multi-Output (MIMO) system, with four inputs and two outputs.

A controller will be designed to asymptotically keep the camera centered on the target, i.e., to reject the target motion disturbance. The target's position in the image plane has to be as close as possible to $(0, 0)$, the center of the image. Image processing algorithms used to compute the target's position have errors due to the noise in the images. The noise is primarily caused by changes in illumination and inexpensive imaging sensor. With vision

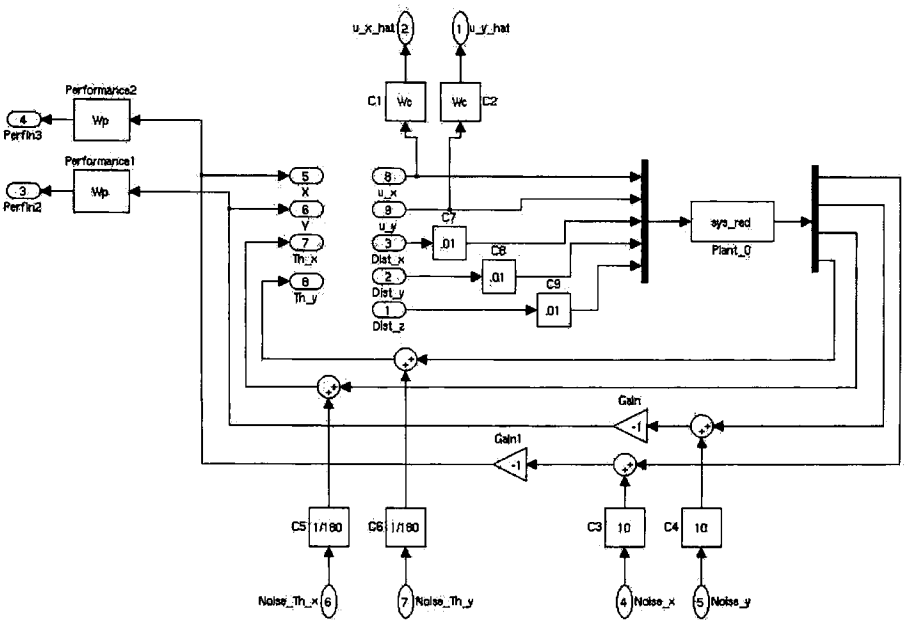



Figure 3.4: Augmented Plant for Robust Controller Synthesis

sensor it is important to have a system with low sensitivity to noise.

### 3.3.1 Initial Design

For the design of the robust controller, input/output (I/O) signals outside of the plant have to be normalzed so that they belong to the unit ball of signals with finite 2-norm. Signals entering the system have to be scaled to the values expected by the plant. Output signals have to be scaled as well. This is accomplished by using input and output weighting filters. The I/O weights could be dynamic or constant, but performance weights and control action weights have to be dynamic to properly represent the system.

Figure 3.4 shows the block diagram used to obtain the augmented plant.

Inputs labeled 1, 2 and 3 in the figure are the target motion disturbance inputs, measured in meters per second. They are scaled with constant input weights.

Inputs 4 and 5 represent the noise from the imaging sensor, measured in pixels. These inputs affect the position of the target in the image plane. They are also scaled with constant

input weights.

Inputs 6 and 7 model the pan and tilt position neasurement noise, measured in degrees. The measurement noise is small and does not have a significant effect on the system, thus a small scaling factor is used.

Finally, inputs 8 and 9 are the control inputs into the plant. These inputs are the reference pan and tilt positions, and are measured in degrees.

The output ports 1 and 2 in Figure 3.4 are the control measurements. Minimization of these outputs places a bound on the control action. It is necessary to minimize the control signal in order to ensure that it does not exceed the actuator limits.

Outputs 3 and 4 are the target's position errors. They are obtained by subtracting the current position of the target in the image from the reference position. Since, the reference position is the center of the image and is $(0, 0)$, the error is obtained by multiplying the current position of the target by $-1$. The ultimate goal of the controller is to minimize these error signals.

Outputs 5, 6, 7 and 8 are the controller inputs. In particular, outputs 5 and 6 are the tracking errors to be minimized by the control algorithm, and outputs 7 and 8 are the current pan and tilt positions of the camera.

The dynamic weights used in the block diagram are:

$$W_p = \frac{0.0033(s + 45000)}{s + 0.1245} \tag{3.17}$$

$$W_c = \frac{1000(s + 0.3)}{s + 300000} \tag{3.18}$$

Bode plots of ${W_p}^{-1}$ and ${W_c}^{-1}$ are presented in Figure 3.5. The performance weight, $W_p$, scales the error signals leaving the system. By adjusting the performance weight, a desired system response can be obtained. In this case, ${W_p}^{-1}$ has a very small gain at low frequencies, which ensures rejection of disturbances that occur at low frequencies. The control weight $W_c$ penalizes the control signal. As seen from Figure 3.5, ${W_c}^{-1}$ is a low pass filter, placing the most of the control effort at low frequencies. It is necessary to shape the control signal in this manner because the actuators in the camera are slow and require

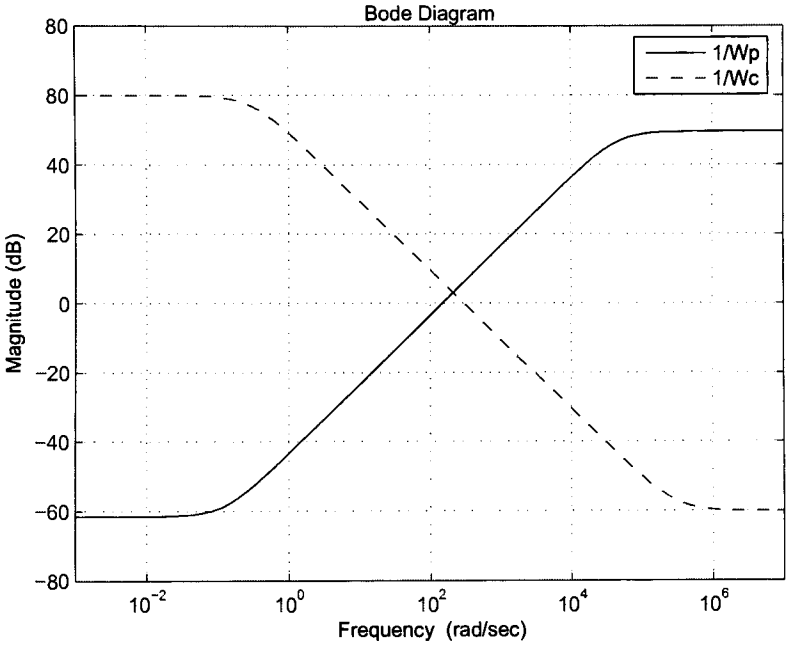



Figure 3.5: Performance and Control Weighting Filters

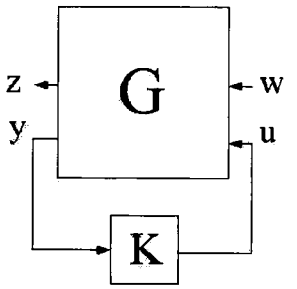



Figure 3.6: Generalized Plant/Controller Interconnection

the control effort to be at lower frequencies.

The plant that the controller is designed for, $Plant\_0$, is the nominal plant extracted from 3.15. It is represented as

$$Plant\_0 = \begin{bmatrix} A & B_2 \\ C_2 & D_{22} \end{bmatrix} \tag{3.19}$$

The generalized plant $G$ was obtained with the help of the Matlab function *linmod*. The system with the generalized plant and the controller is illustrated in figure 3.6.

The input signal $w$ is a vector of three elements — the three disturbance inputs labeled 1, 2 and 3 in Figure 3.4. The control input $u$ has two elements — the two control inputs for pan and tilt labled 8 and 9. The output vector $z$ is the vector of error signals, which have to be minimized. There are four signals to be minimized: the pixel errors and the control inputs. They are labeled 1, 2, 3 and 4 in Figure 3.4. The control output vector, $y$, contains four signals — the 2D position of the target in pixels, labeled 5 and 6 and the camera's pan and tilt position in degrees, labeled 7 and 8.

After the generalized plant $G$ was obtained, an $H_\infty$ controller, $K$, was designed using the *hinfsyn* command from the *$\mu$-analysis and synthesis* Matlab toolbox [5]. This function solves a standard $H_\infty$ problem. It finds a stabilizing controller, $K$, that results in the smallest $H_\infty$ norm, $\gamma$, of the generalized plant, $G$, i.e.,

$$\min_K ||G||_\infty \tag{3.20}$$

Ideally the minimum value of $\gamma$ should be close to 1. But, sometimes it is difficult to design

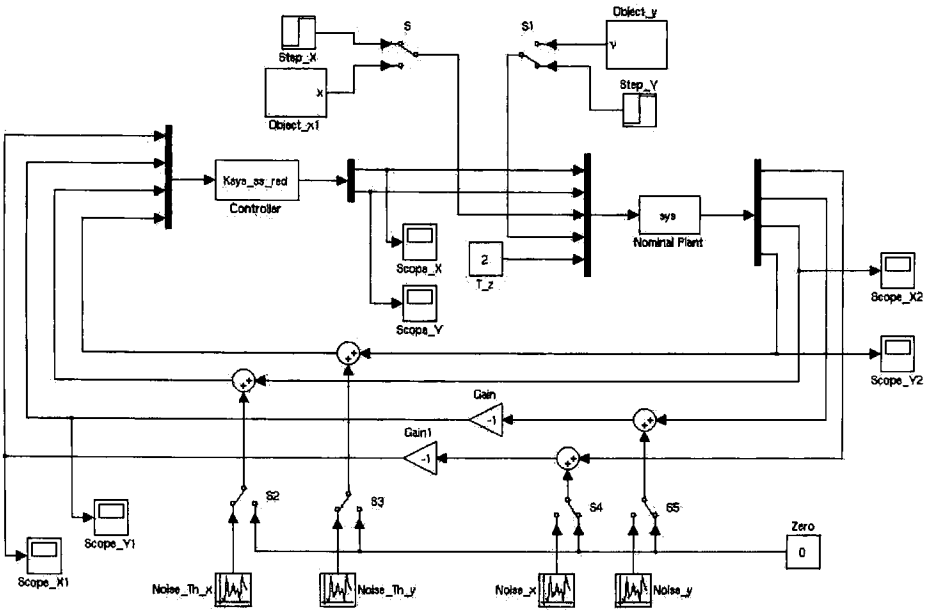



Figure 3.7: Controller Simulation Block Diagram

weighting filters to properly scale the signals. In such cases, when the scaling is not done properly, minimum achievable $\gamma$ may be large. In this case the value of $\gamma$ was about $7 \times 10^6$. With $H_\infty$ design the structure of the uncertainty is not considered. A better measure of the performance of the system can be obtained by computing an upper bound for $\mu$. But the goal of this work is to develop a camera model, thus, this is a preliminary control design and there were made no attempts to lower the value of $\gamma$.

The state-space matrices describing the controller are attached in Appendix D.1. Note, that the controller is a MIMO system with four inputs and two outputs.

### 3.3.2 Simulation

The controller $K$ was simulated with the nominal plant 3.19 and the nonlinear system 2.28.

The simulation block diagram is shown in Figure 3.7. The disturbance input into the plant is a step. The step starts at value of 1 m/s at time 0 and drops to 0 m/s after 0.1

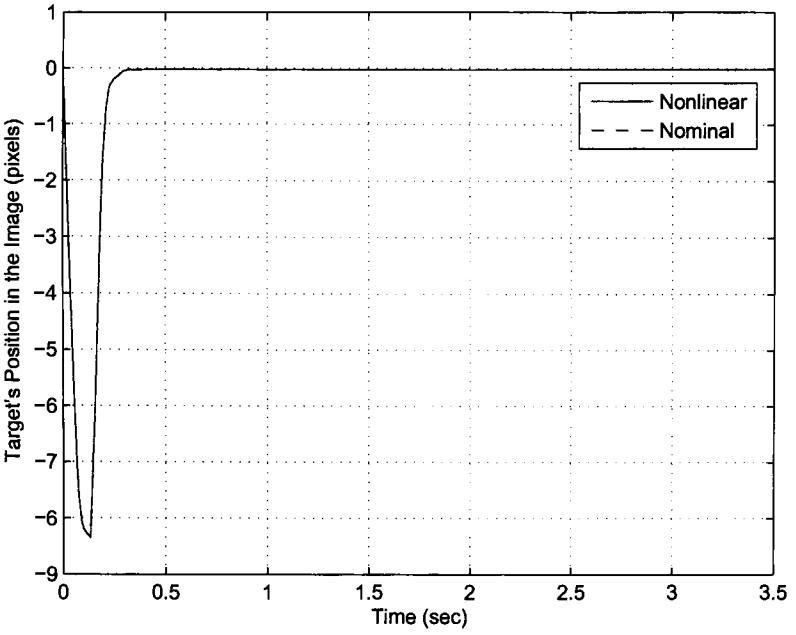



Figure 3.8: Disturbance Rejection with Nonlinear and Nominal Plants

seconds. Since the disturbance input is the velocity, the target moves at a constant velocity at the beginning of the simulation and quickly stops after 0.1 seconds.

The system response is plotted for both nonlinear and nominal plants in Figure 3.8. It can be seen that a very small steady state error is achieved to step disturbances, even though the controller was not designed with an integrator. There is no noticeable difference between the nonlinear and nominal plant simulations. The plot is interpreted as follows. In the beginning of the simulation there is a small delay associated with the physical properties of the camera. Thus, leading to a delay in the response to control action. Then, at about 0.25 seconds, the system catches up with the target and tracks it with an error of less then one pixel.

The discretized version of the controller will be used with the real system. For the discrete-time simulation the controller was discretized using bilinear approximation. It takes 55 ms to extract the location of the target from a single image from the video stream,

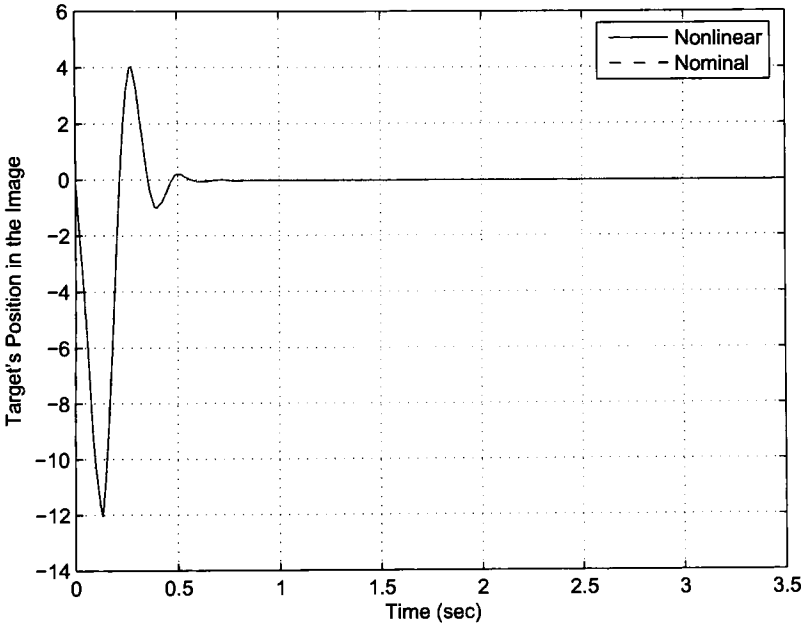



Figure 3.9: Disturbance Rejection with Discretized Controller

thus the discretization was performed with a sampling time of 55 ms to best represent the delays encountered in the real system. State-space matrices for the discretized controller are attached in Appendix D.3. The discrete version of the controller is also simulated with the nominal and nonlinear plants. Results of the simulation are plotted in Figure 3.9. A clear difference in the response between the discretized controller and the continuous one is the presence of oscillations. Now, there is an overshoot of about 4 pixels, right after the initial delay. In simulation, this overshoot is not significant since the maximum value of the overshoot is of the order of the noise in the system. The performance at steady state is very similar to the continuous time case.

### 3.3.3 Refinement of the Design

Unfortunately, due to unmodeled delays and nonlinearities in the system, the performance of the above controller in the experimental system is far below what was expected. The execution time of the image processing algorithms is not constant, it depends on the scene. As a result a 55 ms delay is an approximation of the actual delay, which varies from 45 to 65 ms. A non real-time Operating System (OS) was used in the experiments to control the camera. This means that the assumption of a constant sampling time is, also, an approximation.

The controller designed in the previous section overcompensates resulting in very large overshoots and instability. The control output exceeds the maximum allowable pan and tilt reference angles. In order to solve this problem, the requested performance is reduced. This is done by choosing a different, less aggressive performance weight, $W_p$. Also the contol weight, $W_c$, is changed to reduce the amplitude of the control output. It is expected that decreasing the performance of the controller will decrease overshoots and make the closed loop system stable. The new performance and control weighting filters are:

$$W_p = \frac{0.1(s+1500)}{s+24.75} \tag{3.21}$$

$$W_c = \frac{10(s+30)}{s+3000} \tag{3.22}$$

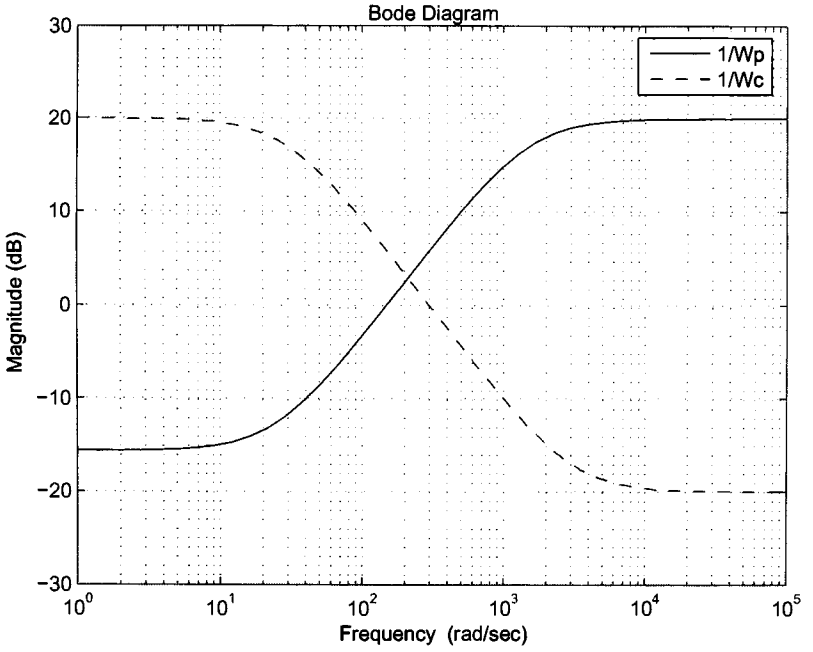



Figure 3.10: Refined Performance and Control Weighting Filters

In Figure 3.10 the bode plots of $W_p^{-1}$ and $W_c^{-1}$ are shown.

Compare to the original weighting filters, in Figure 3.5, the ones selected here have smaller gains at $s = 0$ and $s \to \infty$. Reducing the DC magnitude of $W_p^{-1}$ lessens the disturbance rejection penalty. While, reducing the DC magnitude of $W_c^{-1}$ results in a smaller control effort.

With these new weights the contoller resulting in the system response displayed in Figure 3.11 was synthesized.

It was observed that lower performance controllers that resulted in a reasonable disturbance rejection have large steady state tracking errors, comparable to that in Figure 3.11. In order to eliminate this problem, a controller with an integrator has to be designed. Unfortunately, one of the assumptions of the standard $H_\infty$ problem dictates that the weights be stable. Thus, having an integrator as part of the performance weight is not an option. There are several proposed techniques to allow the design of controllers with unstable weights.

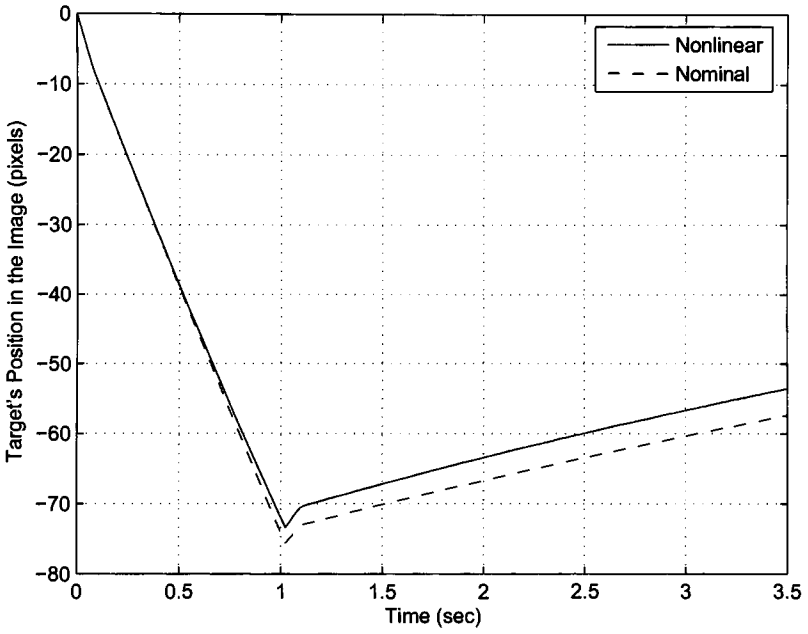



Figure 3.11: Disturbance Rejection with Lower Performance Controller

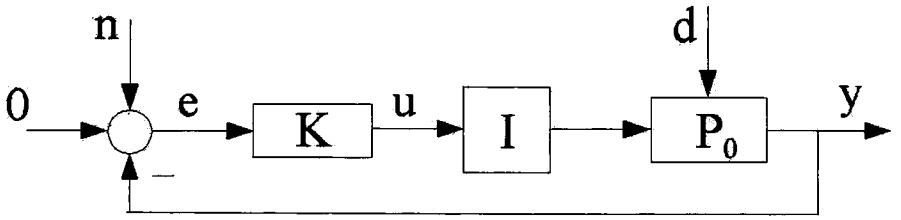



Figure 3.12: Block Diagram of the Model With an Integrator

They are discussed in [28]. Here, a different approach is taken. Integration of the control action is induced by using **relative** pan/tilt positioning of the camera. At each sampling time, the control command moves actuators relative to their current position. This is done by the processor in the internal pan/tilt control of the camera and is transparent to the user. In order to represent this in the simulation, an integrator is added between the controller and the plant. It is important to note that this is just an approximation of relative positioning in the real system. In addition to integration, relative positioning introduces some gain. Thus, in order for the model to be consistent, a gain is incorporated with the integrator. The gain is estimated experimentally, by comparing the results obtained from the actual system and from simulation. The value that was found to be a sufficient approximation is 13.
In order to synthesize an $H_\infty$ controller the integrator with the gain was approximated as:

$$I = \frac{13}{s + 0.0001} \tag{3.23}$$

A block diagram illustrating how the integrator with the gain fits into the overall system is shown in Figure 3.12. Block labeled $P_0$ is the nominal plant, $K$ is the controller and $I$ is the transfer function from 3.23. The input signals into the plant are the integrated control action, $u$, and the disturbnace vector, $d$. The output of the plant, $y$, is subtracted from the reference, which is 0, to produce the error signal, $e$. The error signal is also corrupted by noise $n$.
Controller $K_n$ was synthesized with an integrator and weighting filters from 3.21 and 3.22. The discretezation of the controller was performed using bilinear approximation and 55 ms sampling time, as in the initial design. The state-space matrices for the resulting discrete

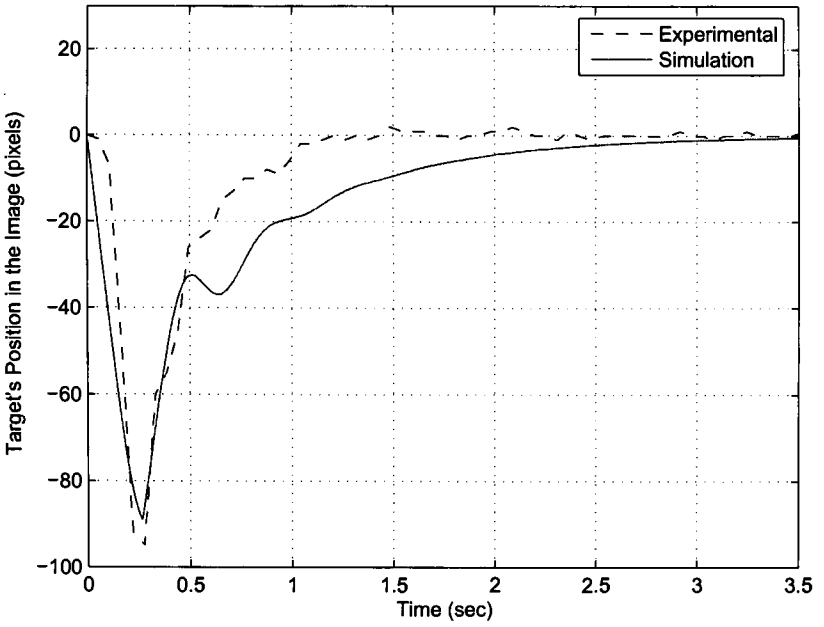



Figure 3.13: Disturbance Rejection with Lower Performance Controller and Integrator

controller, $K_{n_{disc}}$, are attached in Appendix D.2. The simulation and experimental results with the discrete controller are plotted in Figure 3.13.

With this controller, a zero steady state tracking error is achieved. The response of the experimental system has some differences with the response of the simulation. In the simulation there are some oscillations and the settling time is longer than in the experiment. The differences are due to the *Padé* approximation of the delay, which introduces an unstable zero. Overall the camera model gives a good approximation of the experimental system. The model is sufficient for the design of robust controllers with reasonable performance.

### 3.3.4 Controller Order Reduction

With high performance computational systems, the order of the controller may not present a significant problem. With smaller, low power systems, on the other hand, it can cause

problems. Originally, a controller, $K_n$, of order 14 was obtained with the *hinfsyn* function. It was reduced to a controller, $K_{red}$, of order 6 using the Hankel model reduction technique without any noticeable degradation in performance. This means that, for example, when the state matrix $A$ was reduced from 196 to 36 elements, there was a 80% reduction in the required amount of computations.

A function implementing this model reduction method can be found in [5]. The basic idea of this method is to compute Hankel Singular Values (HSV) and remove the states corresponding to the lowest HSVs. As a result, the performance of the controller is expected to degrade. HSVs are computed from the eigenvalues of controllability and observability grammians and represent the "energy" of the states. The states that have relatively low energy have little effect on the system and can be removed.

The plot of Hankel singular values for the controller is presented in Figure 3.14. From the plot it can be seen that the last eight values, $7 - 14$, are much lower then the rest. A new, lower order, controller is obtained by removing the states associated with these eight singular values.

Reducing the order of the controller did not change the results significantly. In Figure 3.15, a simulation of the discretized reduced order controller is compared to that of the original controller. The discretization is performed with the bilinear approximation and sampling time of 55 ms. The state-space equations for the discrete reduced controller, $K_{red_{disc}}$ are attached in Appendix D.4. There are no significant differences between the simulations. The largest discrepancy between the plots is at the part of the responce affected the most by the unstable zero. Experimental results are compared to simulation results in Figure 3.16. Here the results are very similar to the ones in Figure 3.13, where the original controller with 14 states was used.

### 3.3.5 Nominal Plant Selection

The last issue that has to be addressed in modeling single camera systems is the selection of the nominal plant. During the development of the model, a nominal plant was chosen

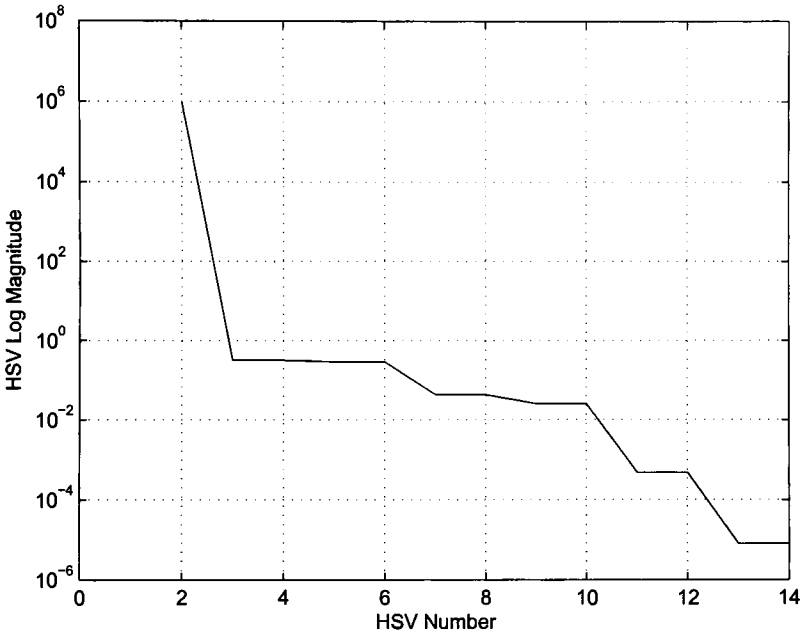



Figure 3.14: Hankel Singular Values of the Controller

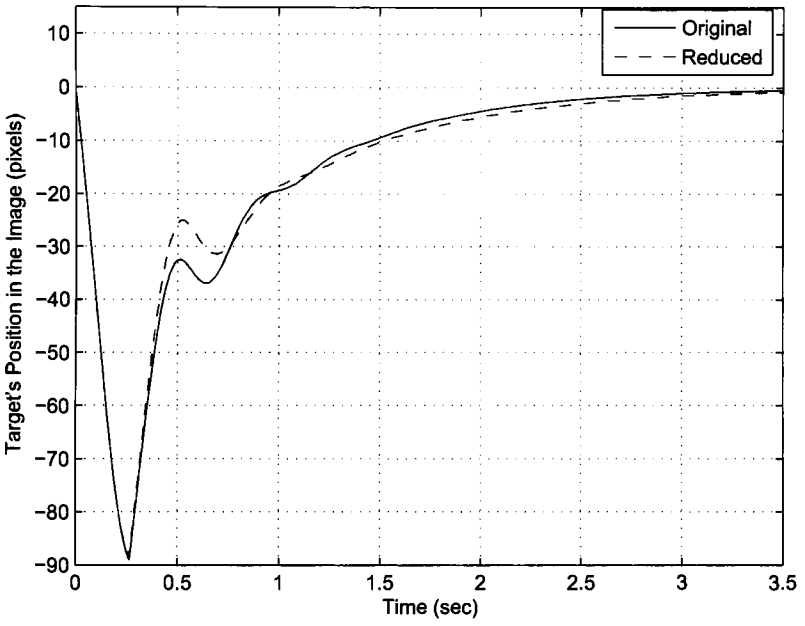



Figure 3.15: Simulation of Reduced and Original Controllers

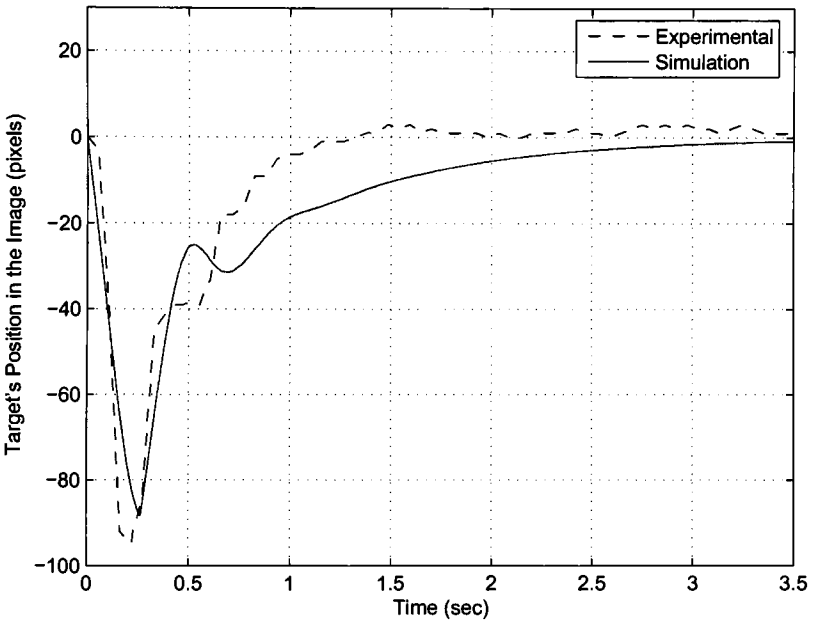



Figure 3.16: Comparison of Experimental and Simulation Results of Reduced Controller

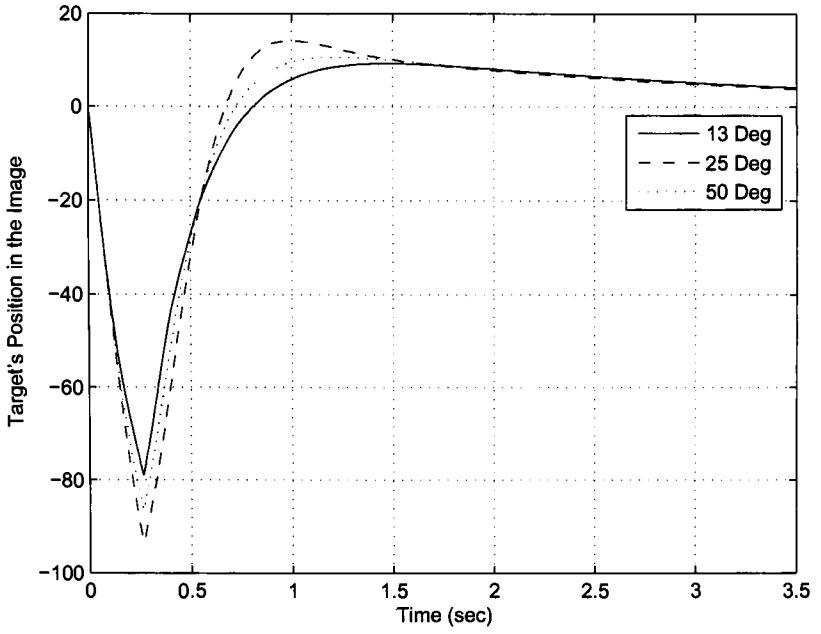



Figure 3.17: Comparison of Different Nominal Plants

to correspond to a $25^o$ pan of the camera. The reasons for making this choice were not discussed earlier. Now there is enough information to address this issue. In practice the choice of the nominal plant for this particular problem does not make a large difference. Responses of three different nominal plants in the closed loop, corresponding to steps of $13^o$, $25^o$ and $50^o$, are plotted in Figure 3.17. As it can be seen from the plots, the difference beetween the three plants is insignificant.

# Chapter 4

# Multicamera Tracking

This chapter outlines the major steps necessary to expand the single camera model introduced in this work to incorporate several cameras. A global coordinate system greatly simplifies the task of multicamera coordination. One approach to establishing a global coordinate system is presented in the following section. Another important aspect is camera positioning and model initialization in the multicamera environment. A section on camera geometry discusses the major initialization steps that need to be taken to expand the single camera model. The final section describes the simulation block diagram of a multicamera system and gives the results of the simulation.

## 4.1 Homogeneous Transformation

With multiple cameras in a 3D environment, each one will have a specific orientation. In order to coordinate multiple cameras, it is necessary to represent the position of the object that they are tracking in a common coordinate system. The common coordinate system will be called a global coordinate system and it will be labeled $XYZ$. Each individual camera also has a local coordinate system, labled $xyz$, associated with it. The origin of the local coordinate system is defined to be at the focal center of the camera. Having a local coordinate system makes it easier to perform local computations.

To transition between the global and the local coordinate systems it is necessary to perform rotation, with respect to each of the three axes, followed by a translation. The necessary

rotations are illustrated in figure 4.1(a). They are:

$$R_z = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

$$R_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{4.2}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \tag{4.3}$$

where, $\alpha$, $\beta$ and $\gamma$ are the respective angles between the local and the global coordinate systems. An example of the rotation along $Y$ axes is illustrated in Figure 4.1(b). Here, the rotation of the original coordinate system, $XYZ$, by $\beta$ radians, along $Y$ axes results in the rotated coordinate system $X'Y'Z'$. In the new coordinate system, $X'Y'Z'$, the $Y'$ axes is aligned with the orignal $Y$ axes and $X'Z'$ plane is rotated by $\beta$ radians with respect to the original $XZ$ plane.

All three rotations can be combined into the following rotation matrix:

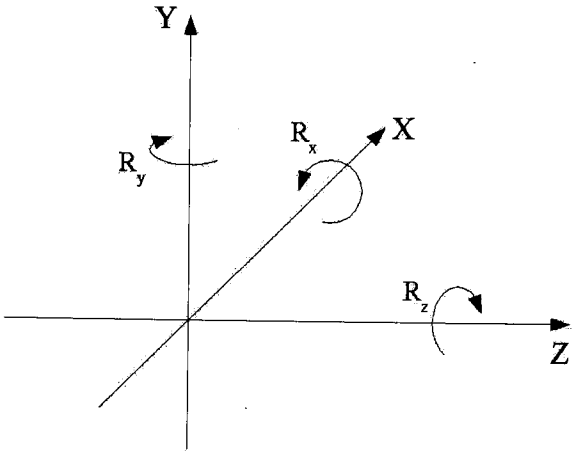
$$R = R_z R_y R_x = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{bmatrix} \tag{4.4}$$

The rotation by matrix $R$ is followed by a translation by vector

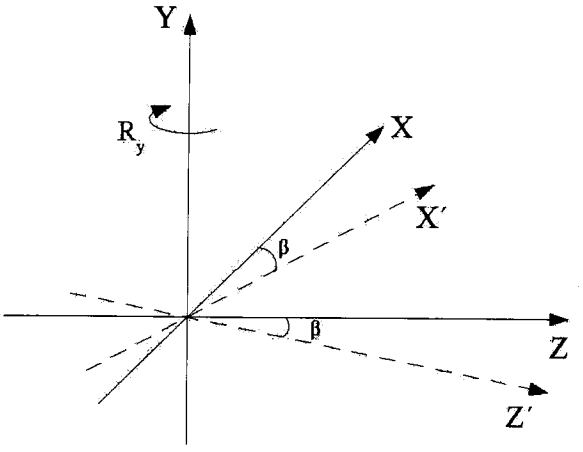
$$t = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{4.5}$$

This is the vector from the origin of the local coordinate system to the origin of the global coordinate system.

Both the rotation and the translation can be described by a single matrix, $T$, called the




(a) Rotations $R_x$, $R_y$ and $R_z$




(b) Rotation along $Y$ axes by $\beta$ radians

Figure 4.1: Axes Rotation

homogeneous transformation matrix, which is:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{4.6}$$

With the homogeneous transformation matrix, the new coordinate can be computed as follows:

$$\tilde{m} = T\tilde{P} \tag{4.7}$$

where $\tilde{P} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}$ is the coordinate in the original coordinate system appended by 1, and $\tilde{m} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}$ is the final coordinate appended by 1.
In a simulation of multiple cameras, each single camera model will have its own coordinate transformation matrix based on its position in the environment.

## 4.2 Multicamera System Geometry

There are infinitely many ways to place cameras in the environment. Only a subset of the configurations will result in a suitable performance of the system. Consider a scenario with three cameras placed as illustrated in Figure 4.2. Assume that the target can enter from the left or from the right of the setup, as illustrated by the arrows. Each camera's coordinates with respect to $XYZ$ axes are labeled as: $C_1 = (X_1, Y_1, Z_1)$, $C_2 = (X_2, Y_2, Z_2)$ and $C_3 = (X_3, Y_3, Z_3)$. In this setup, all of the cameras are located on the $XZ$ plane at $Y = 0$, so that $Y_1 = 0$, $Y_2 = 0$ and $Y_3 = 0$. Cameras at $C_1$ and $C_3$ are aligned with the global coordinate axes, such that $\beta_1 = \beta_3 = 0$, where $\beta$ is the angle of rotation along $Y$ axes, see Equation 4.2. While, the second camera, located at $C_2$, is rotated by $180^0$ resulting in $\beta_2 = \pi$.

The single camera model developed in the earlier sections is updated during the simulation only with the velocities of the camera and the object, not with their positions. For the proper representation of the geometry of the multicamera system, the initial states of the model need to be adjusted with respect to the target's initial location in the environment. Thus, the initial position of the target in the environment, $(x_i, y_i, z_i)$, is required.

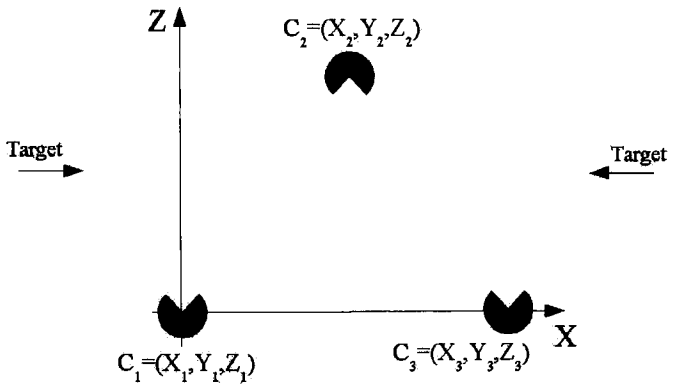



Figure 4.2: Three camera setup

| | |
|---|---|
| $\alpha$ | angle of rotation along Z axes |
| $\beta$ | angle of rotation along Y axes |
| $\gamma$ | angle of rotation along X axes |
| $(X, Y, Z)$ | location of the camera in the global coordinate system |
| $(x, y, z)$ | location of the origin of the global coordinate system in the local coordinates |
| $(x_i, y_i, z_i)$ | initial position of the target in the local coordinates |

Table 4.1: Parameters Required for Initialization

There are several geometric parameters that are required to initialize the simulation of the multicamera system. They are summarized in Table 4.1. The first three parameters, $\alpha$, $\beta$ and $\gamma$, are the angles of rotation along each of the three axes, required to compute the rotation matrix in Equation 4.4. The next parameter is the location of the camera in the global coordinate system, $(X, Y, Z)$. It is used to compute the location of the origin of the global coordinate system in the local coordinates, $(x, y, z)$. The translation vector, $t$, in Equation 4.5, is equal to $(x, y, z)^T$. Finally, the initial position of the target in the local coordinate system, $(x_i, y_i, z_i)$, is required for the initialization of the camera model.
In a real multicamera system these parameters are calculated during the calibration stage. There are several calibration techniques to compute the extrinsic and the intrinsic camera parameters. One such calibration procedure is discussed in great detail in [35].

In application to the three camera scenario described above, three sets of parameters, summarized in Table 4.1, will have to be computed prior to simulation, one set for each

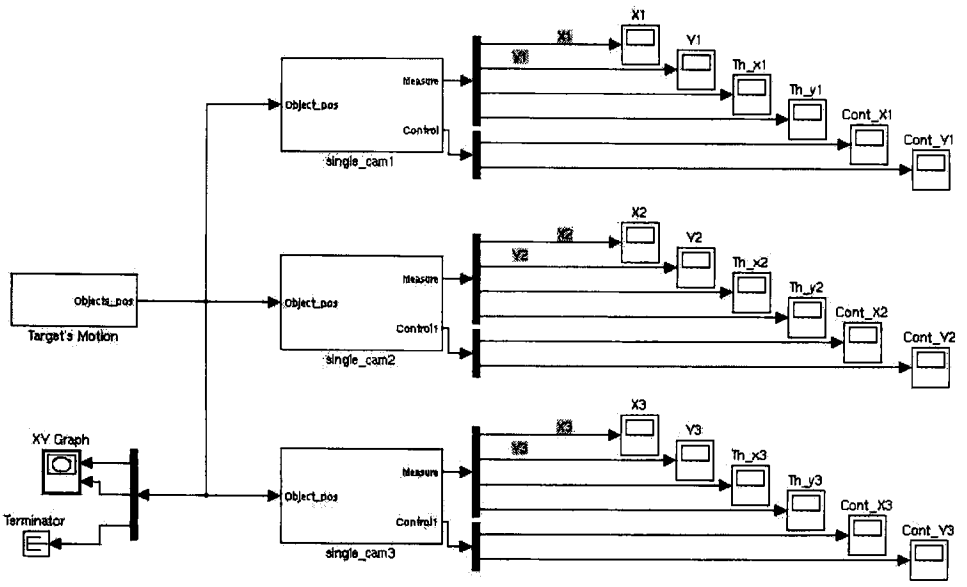



Figure 4.3: Three Camera Simulation Block Diagram

camera.

## 4.3 Simulation of the Multicamera System

The Simulink block diagram of the simulation of the three-camera system, described in the previous section, is illustrated in Figure 4.3.

In the block diagram the *Target's Motion* block, generates the target's position with respect to the global coordinate system. The generated position of the target is the input into the single camera blocks, labeled *single_cam1*, *single_cam2* and *single_cam3*. The single camera blocks are the closed loop tracking subsystems capable of tracking a target from the information extracted from the image.

The block diagram of *single_cam1* is given in Figure 4.4. All of the three *single_cam* blocks are identical with the exception of the constant parameteres passed to the model

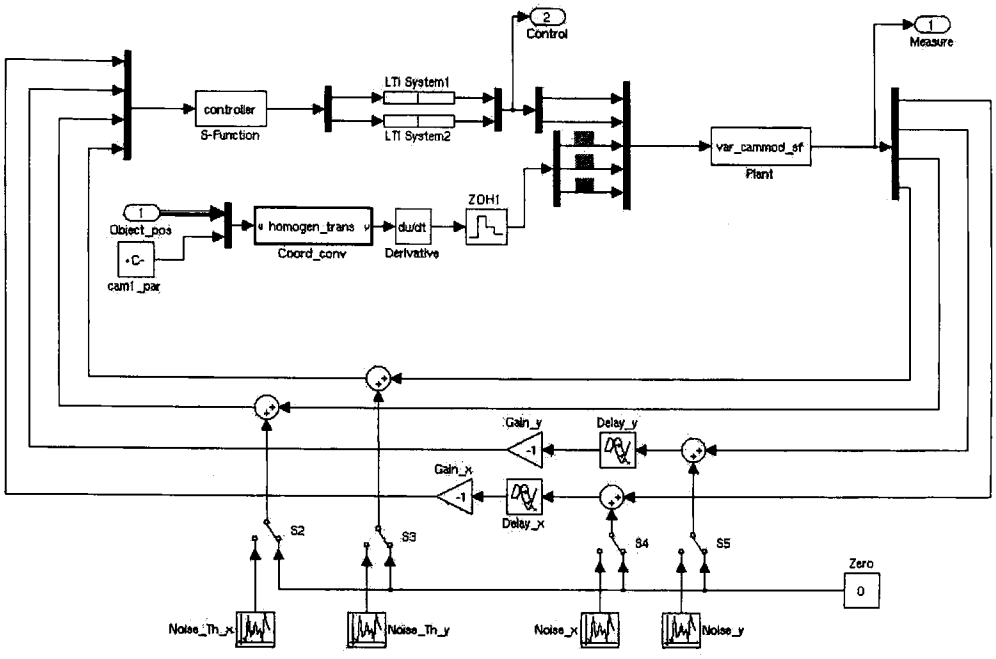



Figure 4.4: Single Camera Tracking Subsystem

for initialization purposes. Each camera will have its own set of parameters listed in Table 4.1. The controller, in Figure 4.4, is the reduced order controller designed previously (see Appendix D.4). The plant is the nonlinear plant represented by the S-Function (see Appendix E.1). The disturbance enters the plant through a block called *Coord_conv*, which implements the rotation matrix 4.4. Since, the plant expects objects velocity as disturbance input, the object's position is differentiated after the coordinate transformation.

Note that only the pan dynamics of the camera are considered during the simulations. Since, the pan and the tilt dynamics are very similar. For this reason the target will move only in the direction of the pan of the camera, *i.e.* in $XZ$ plane. To represent such motion of the target it is sufficient to consider a 1D image.

In this particular simulation the target moves from the left in Figure 4.2, with constant velocity of 0.25 m/s. Table 4.2 lists the parameters used to initialize the simulation. There are three sets of parameters, one for each of the cameras.

| Parameter Name | Parameter Value | Units |
|---|---|---|
| $\alpha_1$ | 0 | radians |
| $\beta_1$ | 0 | radians |
| $\gamma_1$ | 0 | radians |
| $(X_1, Y_1, Z_1)$ | (0,0,0) | meters |
| $(x_1, y_1, z_1)$ | (0,0,0) | meters |
| $(x_{i_1}, y_{i_1}, z_{i_1})$ | (-1,0,1) | meters |
| $\alpha_2$ | 0 | radians |
| $\beta_2$ | $\pi$ | radians |
| $\gamma_2$ | 0 | radians |
| $(X_2, Y_2, Z_2)$ | (0.25,0,2) | meters |
| $(x_2, y_2, z_2)$ | (0.25,0,2) | meters |
| $(x_{i_2}, y_{i_2}, z_{i_2})$ | (1.25,0,1) | meters |
| $\alpha_3$ | 0 | radians |
| $\beta_3$ | 0 | radians |
| $\gamma_3$ | 0 | radians |
| $(X_3, Y_3, Z_3)$ | (0.5,0,0) | meters |
| $(x_3, y_3, z_3)$ | (-0.5,0,0) | meters |
| $(x_{i_3}, y_{i_3}, z_{i_3})$ | (-1.5,0,1) | meters |

Table 4.2: Parameters Required for Initialization

The results of the simulation are presented in Figure 4.5. For each of the cameras the target's position in the image, in pixels, is plotted against the simulation time. In all three cases there is a steady state tracking error of about 28 pixels. This is due to the fact that the input is a ramp. The controller was designed to reject step disturbances. Thus, some steady state tracking error is expected with ramp disturbances. This is discussed in more detail in the next section.

The target tracking output of camera 1 is shown at the top of Figure 4.5. The initial position of the object in the image is at -200 pixels. It is negative, because the object starts moving to the left of the first camera. Therefore, the object is projected to the left portion of the image, which has negative pixel values assigned to it.

The target tracking output of camera 2 is shown in the middle of Figure 4.5. Here, the initial position of the object is 250 pixels. It is positive, since the object starts its motion to the right of the second camera. The second camera is located 0.25 meters further from

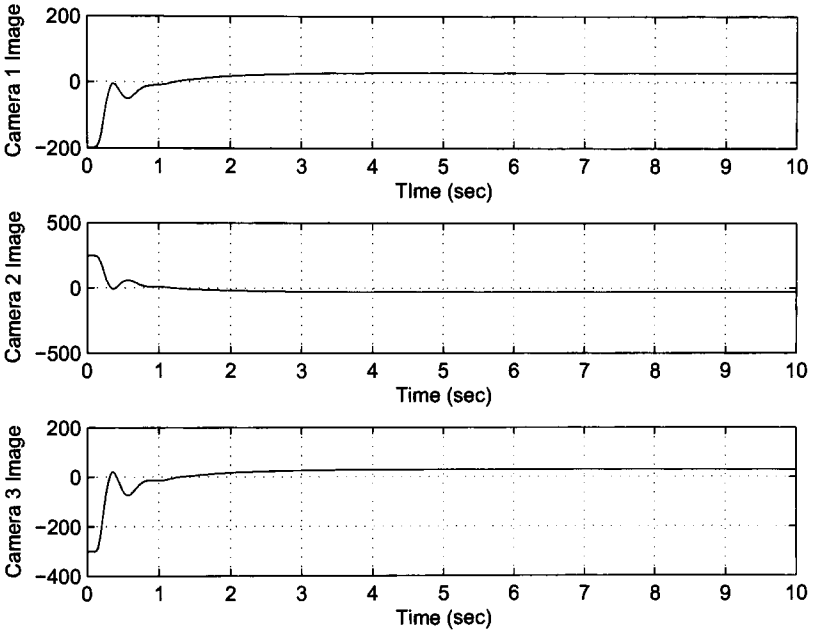



Figure 4.5: Target Tracking in a Multicamera System

the object's initial position then the first camera. So, the object starts its motion 50 pixels further from the origin of the image then in the case of the first camera.

Finally, the target tracking output of camera 3 is shown at the bottom of Figure 4.5. The initial position of the object is -300 pixels. Again, as with the first camera, the object starts its motion to the left of the third camera, so the position is negative. And the third camera is located 0.25 meters further from the object's initial position then the second camera. Thus, as expected, the object starts its motion 50 pixels further from the origin then in the case of the second camera.

This was a simple simulation where the fields of views of the cameras mostly overlapped. Also, all of the cameras started to track the object at the very begining of the simulation, since physical constaints of the image sensor, such as finite image size, were not considered.

## 4.4 Steady State Tracking Error

It was mentioned eralier that the controller can not reject ramp disturbances with zero steady state tracking error. This is because only a simple integrator is present in the system. With such a system, zero steady state tracking error is achievable only for step disturbances. The amount of error at steady state depends on the slope of the disturbance ramp function. In other words, the velocity of the target determines how much of the steady state error is present in the system. To illustrate this a response of one of the cameras is plotted for different velocities of the target in Figure 4.6. The solid line corresponds to the velocity of 0.25 m/sec, and the dashed line corresponds to the velocity of 0.5 m/sec. It is clear from the figure that, as expected, a higher velocity of the target corresponds to a larger error in the steady state response of the system.

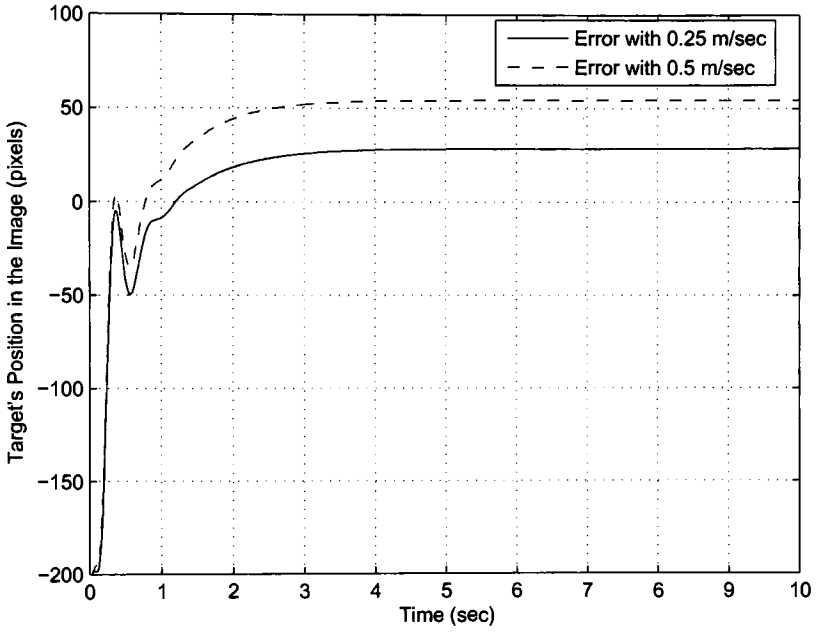



Figure 4.6: Steady State Tracking Error with Different Velocities of the Target

## 4.5 Constrained Multicamera System Simulation

There are two main sources of saturation in the physical camera system. One comes from the pan/tilt actuator limits. And the other is a result of the finite size of the image sensor. In the previous section these saturation effects were not considered. In particular, all of the cameras could pan and tilt to any angular position and the image was projected onto a plane without considering the size of the image sensor. In order for the simulation to be more realistic, these issues have to be addressed.

Consider a camera with pan limit of $n$ degrees. So that the achievable pan position is bound by the interval $[-n, n]$. Furthermore, the image produced by the camera is $k$ pixels wide. The target is visible by the camera only if its image coordinate is bound by the interval: $[-k/2, k/2]$; where point with coordinate $(0)$ is the center of the image.

The way the saturation is introduced into the simulation has to be carefully considered. A simple approach is to saturate the controller output, so that it does not try to move the camera beyond the actuator pan limit. If, at the same time, the position of the target in the image is saturated at the image borders, inconsistent results may be obtained.

This is illustrated by the simulation output in Figure 4.8. Controller output is saturated at 25 degrees, so that $n = 25$ and the image size is set to 100 pixels, so that $k = 100$. Figure 4.8(b) shows the control action before it is saturated. The target starts its motion beyond the field of view of the camera. But, since the target is being continuously projected onto the image plain, there is some error between the desired and the actual positions of the target. The contoller tries to minimize this error and moves the camera in the direction of the target. Obviously, this behavior is inconsistent with an actual multicamera system — the target is outside the field of view of the camera.

Also, it can be seen that the controller attempts to move the camera beyond -25 degrees, the saturation limit. When the target crosses the origin of the image at about 2.5 seconds of the simulation, Figure 4.8(a), there is some delay before the control action has an effect on the system. The delay is contributed by the fact that the controller is moving back from the saturated region until just after 4 seconds of simulation.

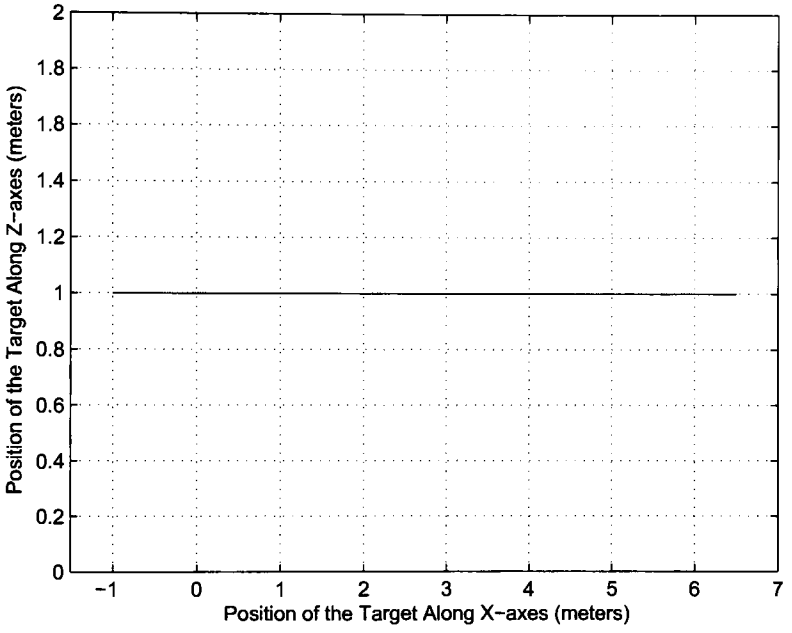
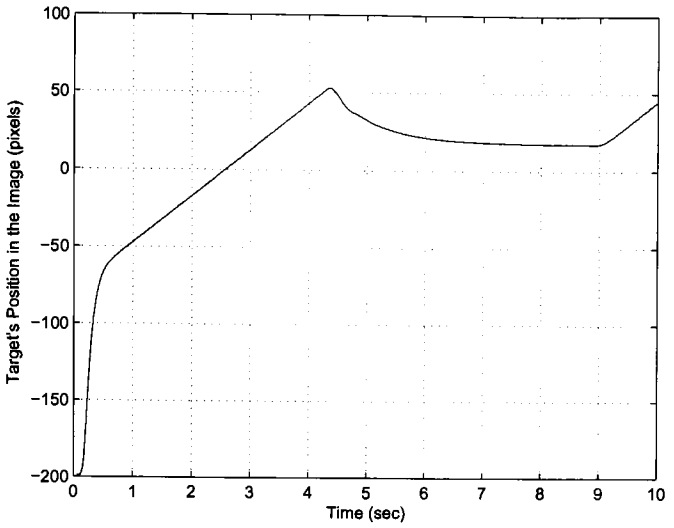
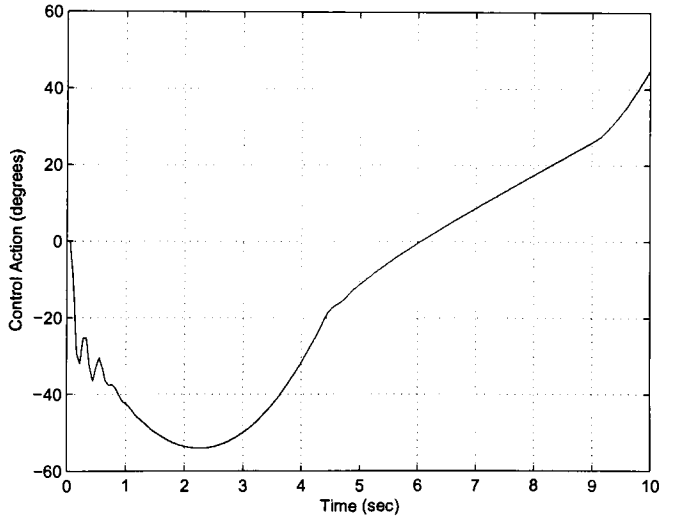



Figure 4.7: Trajectory of the Target




(a) Target's Motion in the Image




(b) Controller output

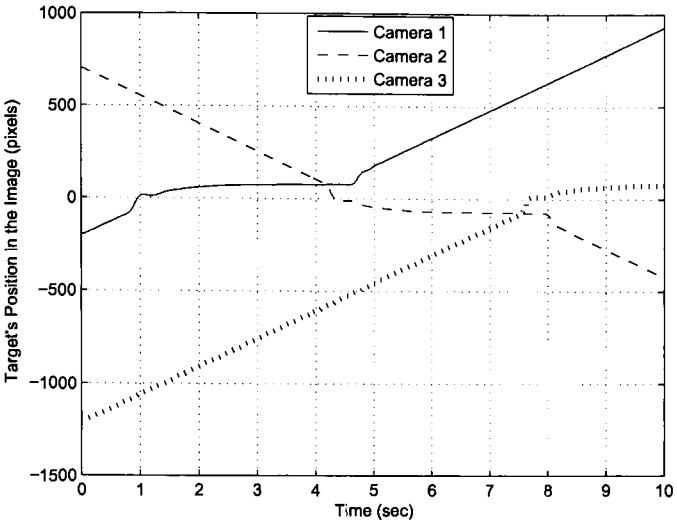
Figure 4.8: Improper Model Saturation Example

| Parameter Name | Parameter Value | Units |
|---|---|---|
| $\alpha_1$ | 0 | radians |
| $\beta_1$ | 0 | radians |
| $\gamma_1$ | 0 | radians |
| $(X_1, Y_1, Z_1)$ | (0,0,0) | meters |
| $(x_1, y_1, z_1)$ | (0,0,0) | meters |
| $(x_{i_1}, y_{i_1}, z_{i_1})$ | (-1,0,1) | meters |
| $\alpha_2$ | 0 | radians |
| $\beta_2$ | $\pi$ | radians |
| $\gamma_2$ | 0 | radians |
| $(X_2, Y_2, Z_2)$ | (2.5,0,2) | meters |
| $(x_2, y_2, z_2)$ | (2.5,0,2) | meters |
| $(x_{i_2}, y_{i_2}, z_{i_2})$ | (3.5,0,1) | meters |
| $\alpha_3$ | 0 | radians |
| $\beta_3$ | 0 | radians |
| $\gamma_3$ | 0 | radians |
| $(X_3, Y_3, Z_3)$ | (5,0,0) | meters |
| $(x_3, y_3, z_3)$ | (-5,0,0) | meters |
| $(x_{i_3}, y_{i_3}, z_{i_3})$ | (-6,0,1) | meters |

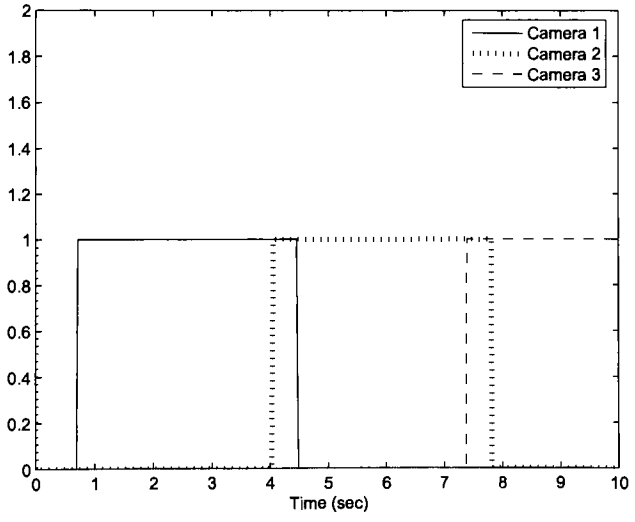
Table 4.3: Parameters Required for Initialization

A different approach to implement the saturation effects is to turn off the controller when the target is outside of the image or when the pan limits are reached. This approach takes care of both, the actuator saturation and the image saturation. In order to turn off the controller the error signals fed to the controller are set to zero. This forces the control output to go to zero. Since the control output is integrated before entering the plant, the control signals entering the plant do not change. As a result, when the target is outside of the field of view of the camera or the pan limit is reached, the camera does not move.

Note that it is necessary to always update the camera model with the current object's velocity to avoid reinitialization of the parameters during the simulation. This means that the position of the object in the image is always computed, even when the target is outside of the field of view of the camera. Of course, the information about the target outside of the image borders is not relevant to the rest of the simulation. It can not be used to determine where the target is because it is unknown in an actual system.




(a) Line Tracking With Saturation




(b) Controller State On/Off

Figure 4.9: Tracking of an Object Moving on a Straight Line

An output of the simulation with such implementation of saturation and initialization parameters from Table 4.3 is shown in Figure 4.9. In Figure 4.9(a) the image coordinates of the target from all three cameras are combined into one plot. The target enters the field of view of camera 1 at 0.8 seconds of simulation. Then it is tracked with a contstant error, by panning the camera, until 4.5 seconds of simulation. At this point the target exists the field of view of camera 1, and camera 1 is at its pan limit of 100 degrees. But, before the target exits the field of view of camera 1 it moves into the field of view of camera 2. The cameras are positioned in such a way, that only a little portion of their respective fields of view overlap. In this way a target can be tracked over a larger area.

Figure 4.9(b) shows the times at which each camera's controller is on and off. A controller is turned on, when the target enters the field of view of the respective camera. Here it is clear that the target is continuously tracked by at least one camera starting at 0.8 secods of the simulation time.

The simulation in this section is more realistic than the one without actuator saturation. It enables the simulation of systems with non overlaping or just slightly overlapping fields of view.

## 4.6 Simulation With Variable Depth

So far, all of the simulations were done with the target moving in a straight line, without changes in depth. The camera model in its current implementation does not allow variable distance from the camera to the target. This, obviously, adds a lot of limitations to what can be done in the simulation. For example, when the target moves in a sinusoid trajectory, as illustrated in Figure 4.10, the resulting simulation output, Figure 4.11, is exactly the same as the output of the simulation with the target moving in a straight line, Figure 4.9.

Without consideration of the variable depth in the model it is very difficult to combine information from multiple cameras. The position of the target in the images of different cameras is inconsistent.

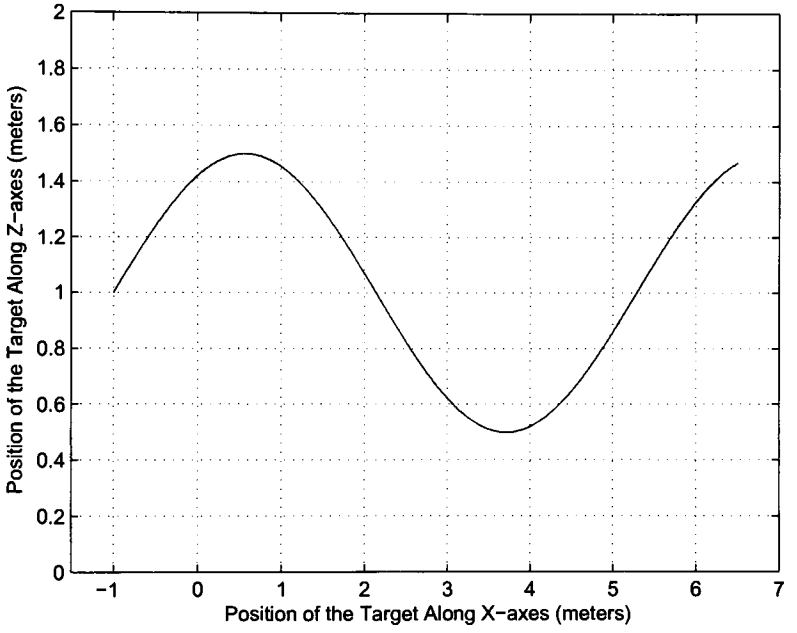
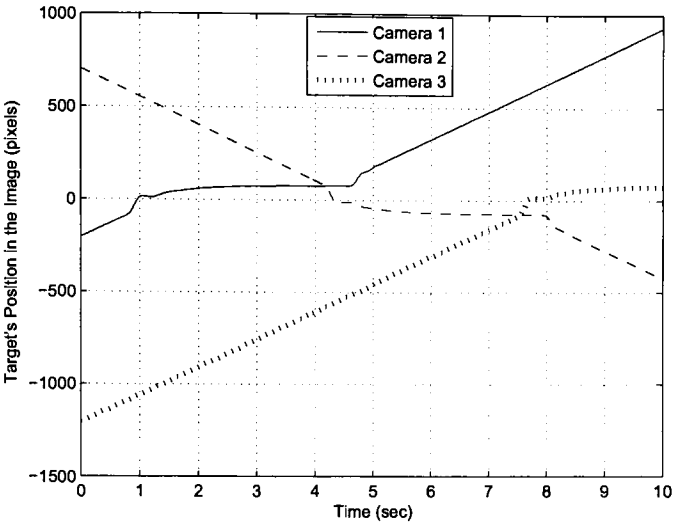
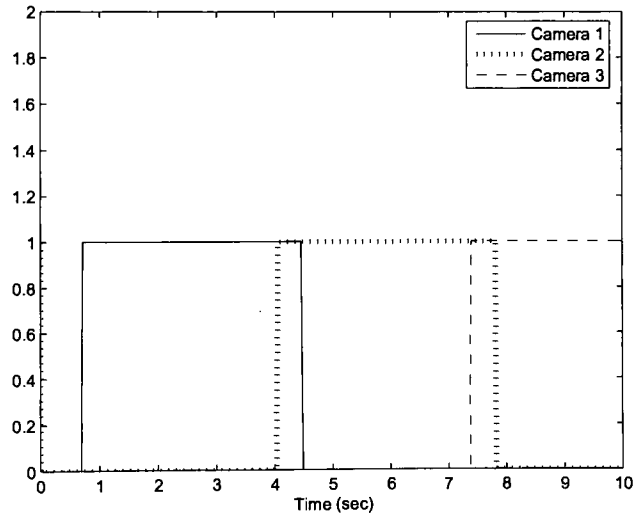



Figure 4.10: Trajectory of the Target




(a) Variable Depth Tracking With Saturation




(b) Controller State On/Off

Figure 4.11: Tracking of an Object With Variable Depth Motion

The simulation discussed here can easily be expanded to any number of cameras by adding more single camera blocks and their respective initialization parameters. At this point, single camera subsystems were controlled independently via local controllers. The resulting tracking performance of each camera in this system is not better then that of a single camera. The next step would be to expand the model to incorporate the variable depth and to design a controller, that incorporates the information from all of the cameras, to potentially improve the tracking performance.

# Chapter 5

# Conclusions and Future Work

In this work, a nonlinear single camera model combining the kinematic relationships of the pixel motion in the image with respect to the motions of the target and the camera, and the dynamic properties of the pan and tilt of a camera was introduced. The model was developed for an off-the-shelf pan/tilt camera. The camera has many nonlinearities which were represented as uncertain parameters in a linear fractional representation of the system with the uncertainty block. All of the nonlinear parameters were mapped to uncertain ones in the linear fractional representation, resulting in a linear uncertain model. The model was verified against experimental response data obtained on a real system. It was shown that there are no significant differences between the two.

A controller was designed for nominal performance using $H_\infty$ optimal control. The resulting controller had nominal tracking performance of a target with respect to step disturbances and noise that enters the system through image processing algorithms. The controller was verified in the experimental system with a SONY EVI-D100 pan/tilt camera. The robust control framework is, indeed, a suitable approach to the contoller design for active vision systems.

The main advantage of using off-the-shelf pan/tilt camera as opposed to high performance linear pan/tilt platform is the cost. The trade off is longer design and implementation times. The choice of particular hardware is application dependent, but in general for mass produced systems component costs play a significant role. Here, it was shown that modern

control systems tools are capable of handling various uncertainties and nonlinearities inherent to the lower priced sensors and actuators.

This work is the first step in the development of a multicamera system model. The necessary steps to expand a single camera model to a multicamera one were discussed in the last chapter. The results of the simulation of a simple multicamera system were also presented. The proposed approach to multicamera system modeling is scalable and easy to implement.

The next step is to develop a method for controller synthesis for multicamera systems. There are several ways to approach controller synthesis. One, is to have each camera track the target independently, as was done here. This is a completely distributed approach. This method is simple — a single camera controller as the one designed here was sufficient. An alternative would be to design a controller that uses information from all of the cameras to produce control action. This is the centralized control approach. This way the control will be based on the estimated 3D location of the object, which might improve the performance of the system. There is also the possibility to design a hierarchical control system. These are topics that merit further research.

# Bibliography


[1] S. Araki, T. Matsuoka, H. Takemura, and N. Yokoya. Real-time tracking of multiple moving objects in moving camera image sequences using robust statistics. In *IEEE Fourteenth International Conference on Pattern Recognition*, volume 2, pages 1433–1435, August 1998.

[2] Ronald C. Arkin. *Behavior-Based Robotics*. Bradford Books, 1998.

[3] Castaneda B., Luzanov Y., and Cockburn J.C. A modular architecture for real-time feature-based tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2004.

[4] Gary Balas, Richard Chiang, Andy Packard, and Michael Safonov. *Robust Control Toolbox For Use with MATLAB*. The MathWorks Inc., 3.0.2 edition, 2005.

[5] Gary J. Balas, John C. Doyle, Keith Glover, Andy Packard, and Roy Smith. *$\mu$-Analysis and Synthesis Toolbox For Use with MATLAB*.

[6] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[7] Michael S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, April 1998.

[8] J. Chen, D.M. Dawson, W.E. Dixon, and V.K. Chitrakaran. Navigation function based visual servo control. In *American Control Conference*, volume 5, pages 3682–3687, June 2005.

[9] J.C. Cockburn and B.G. Morton. Linear fractional representations of uncertain systems. In *Automatica*, volume 33, pages 1263–1271, 1997.

[10] Juan C. Cockburn. Linear fractional representation of systems with rational uncertanty. In *ACC*, volume 2, pages 1008–1012, June 1998.

[11] Juan C. Cockburn. Multidimensional realizations of systems with parametric uncertainty. In *MTNS*, 2000.

[12] Robert T. Collins, Alan J. Lipton, Hironobu Fujiyoshi, and Takeo Kanade. Algorithms for cooperative multisensor surveillance. *IEEE*, 89(10):1456–1477, October 2001.

[13] R. D'Andrea and S. Khatri. Kalman decomposition of linear fractional transformation representations and minimality. In *American Control Conference*, volume 6, pages 3557–3561, June 1997.

[14] Raymond A. Decarlo, Michael S. Branicky, Stefan Pettersson, and Bengt Lennartson. Perspectives and results on the stablity and stabilizability of hybrid systems. *IEEE*, 2000.

[15] Shilon L. Dockstader and A. Murat Tekalp. Multiple camera fusion for multi-object tracking. In *2001 IEEE Workshop*, pages 95–102. University of Rochester, July 2001.

[16] John Dorsey. *Continuous and Discrete Control Systems: Modeling, Identification, Design and Implementation.* McGraw-Hill College, 2001.

[17] Magnus Egerstedt and Xiaoming Hu. A hybrid control approach to action coordination for mobile robots. *Automatica*, 38:125–130, 2002.

[18] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeni. *Feedback Control of Dynamic Systems*. Prentice Hall, 4 edition, December 2002.

[19] B. Heisele and C. Woehler. Motion-based recognition of pedestrians. In *IEEE Fourteenth International Conference on Pattern Recognition*, volume 2, pages 1325–1330, August 1998.

[20] Roberto Horowitz and Pravin Varaiya. Control design of an automated highway system. *IEEE*, 2000.

[21] Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, and Shankar Sastry. Regularization of zeno hybrid automata. *Syst. Control Lett.*, 38, 1999.

[22] Mikael Johansson and Anders Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, April 1998.

[23] John Lygeros. Lecture notes on hybrid systems. 2004.

[24] John Lygeros, Karl Henrik Johansson, Slobodan N. Simic, Jun Zhang, and S. Shankar Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48:2–17, January 2003.

[25] John Lygeros, Shankar Sastry, and Claire Tomlin. The art of hybrid systems. July 2001.

[26] J.-F. MAGNI. *Linear Fractional Representation Toolbox. Modelling, Order Reduction, Gain Scheduling*. ONERA Systems Control and Flight Dynamics Department, July 2004.

[27] Judit Martinez, Eva Costa, Paco Herreros, Xavi Sanches, and Ramon Baldrich. A modular and scalable architecture for pc-based real-time vision systems. *Real-Time Imaging*, 9:99–112, January 2003.

[28] Gjerrit Meinsma. $\mathcal{H}_\infty$ control with unstable and nonproper weights.

[29] Katja Nummiaro, Esther Koller-Meier, Tomas Svoboda, Daniel Roth, and Luc Van Gool. Color-based object tracking in multicamera environments. *DAGM*, pages 591–599, September 2003.

[30] N. Papanikolopoulos. Integrating computer vision and control for vision-assisted robotic tasks. *Proceedings of the American Control Conference*, 1:904–908, June 1995.

[31] N. Papanikolopoulos, Pradeep Khosla, and Takeo Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, February 1993.

[32] P. A. Parrilo, M. Sznaier, R. S. Snchez Peac, and T. Inanc. Mixed time/frequency-domain based robust identification. *Automatica*, 34(11):1375–1389, November 1998.

[33] Y. Sheikh and Mubarak Shah. Bayesian object detection in dynamic scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 74–79, June 2005.

[34] Mario Sznaier, Tamer Inanc, and Octavia Camps. Robust controller design for active vision systems. In *American Control Conference*, volume 3, pages 2013–2017, June 2000.

[35] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[36] M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice Hall, second edition, 1993.

[37] Hui Ye, Anthony N. Michel, and Ling Hou. Stability theory for hybrid dynamical systems. *IEEE Transactions on Automatic Control*, 43(4):461–474, April 1998.

[38] Kemin Zhou and John C. Doyle. *Essentials of Robust Control*. Prentice Hall, 1997.

# Appendix A

# Summary of the State-Space Equations

$$
\begin{aligned}
\dot{\mathbf{z}} &= A(\mathbf{z})\mathbf{z} + B\mathbf{u} + E(\mathbf{z})\mathbf{v} \\
\mathbf{y} &= C\mathbf{z} + \eta
\end{aligned}
$$

$$
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{p}_{1x} \\ \dot{p}_{2x} \\ \dot{p}_{3x} \\ \dot{p}_{1y} \\ \dot{p}_{2y} \\ \dot{p}_{3y} \end{bmatrix} =
\begin{bmatrix}
0 & 0 & c_{21x}xyn_y & c_{22x}xyn_y & 0 & -c_{21y}(\frac{1}{n_x} + x^2 n_x) & -c_{22y}(\frac{1}{n_x} + x^2 n_x) & 0 \\
0 & 0 & c_{21x}(\frac{1}{n_y} + y^2 n_y) & c_{22x}(\frac{1}{n_y} + y^2 n_y) & 0 & -c_{21y}xyn_x & -c_{22y}xyn_x & 0 \\
0 & 0 & a_{11x} & a_{12x} & a_{13x} & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & a_{11y} & a_{12y} & a_{13y} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix} x \\ y \\ p_{1x} \\ p_{2x} \\ p_{3x} \\ p_{1y} \\ p_{2y} \\ p_{3y} \end{bmatrix} +
\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}
\begin{bmatrix} \Theta_{rx} \\ \Theta_{ry} \end{bmatrix} +
\begin{bmatrix}
\frac{1}{Z_s n_x} & 0 & -\frac{x}{Z_s} \\
0 & \frac{1}{Z_s n_y} & -\frac{y}{Z_s} \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}
$$

$$
\begin{bmatrix} x \\ y \\ \Theta_x \\ \Theta_y \end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c_{12x} & c_{13x} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & c_{12y} & c_{13y}
\end{bmatrix}
\begin{bmatrix} x \\ y \\ p_{1x} \\ p_{2x} \\ p_{3x} \\ p_{1y} \\ p_{2y} \\ p_{3y} \end{bmatrix} + \eta
$$

# Appendix B

# Symbolic LFR

$$M_{11} = \begin{bmatrix} 0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ O_{26\times32} \end{bmatrix} \tag{B.1}$$

$$M_{21} = \begin{bmatrix} O_{2\times32} \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ O_{2\times32} \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ O_{2\times32} \end{bmatrix} \tag{B.2}$$

$$
M_{12} = \left[\begin{array}{cccccccccccc}
\multicolumn{12}{c}{O_{6\times 13}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_2k_5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_2k_5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_4k_5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_4k_5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_2k_5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_2k_5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_1k_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_1k_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_3k_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_3k_5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_1k_5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k_1k_5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & k_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & k_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right] \tag{B.3}
$$

$$M_{22} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.4}$$

$$\Delta(q) = diag \left[ I_3 q_1 \;\; I_3 q_2 \;\; q_3 \;\; q_4 \;\; q_5 \;\; q_6 \;\; q_7 \;\; q_8 \;\; I_3 q_9 \;\; I_3 q_{10} \;\; I_3 q_{11} \;\; I_3 q_{12} \;\; q_{13} \;\; q_{14} \;\; q_{15} \;\; q_{16} \;\; I_4 q_{17} \right] \tag{B.5}$$

# Appendix C

# Reduced LFR Model

The following are $L_{red}$ and $\Delta_{red}$ matrices comprising the reduced LFR model. Matrix $L_{red}$ is divided into the following blocks:

$$L_{red} = \left[\begin{array}{c|c} L_{11} & L_{12} \\ \hline L_{21} & L_{22} \end{array}\right] = \left[\begin{array}{cc|c} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ \hline C_2 & D_{21} & D_{22} \end{array}\right] \tag{C.1}$$

$$\Delta_{red}(\delta) = diag\left[ I_8 \tfrac{1}{s}\ I_2\delta_1\ I_2\delta_2\ I_1\delta_3\ I_1\delta_4\ I_1\delta_5\ I_1\delta_6\ I_1\delta_7\ I_1\delta_8\ I_1\delta_9\ I_1\delta_{10}\ I_1\delta_{11}\ I_1\delta_{12}\ I_1\delta_{13}\ I_1\delta_{14}\ I_1\delta_{15}\ I_1\delta_{16}\ I_3\delta_{17} \right] \tag{C.2}$$

$$A = \begin{bmatrix} -1088.5 & 0 & 64757 & 0 & 0\ 0 & 6.6882e+05 & 0 \\ 0 & -1088.5 & 0 & 64757 & 0\ 0 & 0 & -6.6882e+05 \\ -1.6204 & 0 & 36.909 & 0 & 0\ 0 & 381.2 & 0 \\ 0 & -1.6204 & 0 & 36.909 & 0\ 0 & 0 & -381.2 \\ 0 & 0 & 0 & 2.3365e+06 & 0\ 0 & 0 & 1331.7 \\ 0 & 0 & 2.3365e+06 & 0 & 0\ 0 & -1331.7 & 0 \\ 0 & 0 & 1 & 0 & 0\ 0 & -0.00056996 & 0 \\ 0 & 0 & 0 & -1 & 0\ 0 & 0 & -0.00056996 \end{bmatrix} \tag{C.3}$$

$$B_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -323700 & -20433000 & -2.133e+08 & 0 & 0 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 323700 & 20433000 & 2.133e+08 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -184.49 & -11646 & -121570 & 0 & 0 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 184.49 & 11646 & 121570 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1599 & 0 & 0 & 0 & 0 & 0 & 0 & 91749 & 213450000 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & -519.67 & 0 \\ 0 & -1599 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 91749 & 213450000 & 0\ 0\ 0\ 0\ 0 & -519.67 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\ 0\ 0\ 0\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.4}$$

$$B_2 = \begin{bmatrix} 0 & -319.79 & 0 & 0 & 0 \\ 319.79 & 0 & 0 & 0 & 0 \\ 0 & -0.18227 & 0 & 0 & 0 \\ 0.18227 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2093.1 & 0 \\ 0 & 0 & -2093.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.5}$$

$$
C_1 = \left[\begin{array}{cccccccc}
0 & 0 & 0.25831 & 0 & 0 & 0 & -0.00014723 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.25831 & 0 & 0 & 0 & 0.00014723 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.003127 & 0 & 1.7823e-06 & 0 & 0 & 0 & 0 \\
0 & 1.7823e-06 & 0 & -0.003127 & 0 & 0 & 0 & -1.7823e-06 \\
0 & 1.0158e-09 & 0 & -1.7823e-06 & 0 & 0 & 0 & 0.003127 \\
-0.003127 & 0 & -1.7823e-06 & 0 & 0 & 0 & 0 & 0 \\
-1.7823e-06 & 0 & 0.003127 & 0 & 0 & 0 & -1.7823e-06 & 0 \\
-1.0158e-09 & 0 & 1.7823e-06 & 0 & 0 & 0 & 0.003127 & 0 \\
0 & -0.010915 & 0 & -6.2213e-06 & 0 & 0 & 0 & 0 \\
0 & -6.2213e-06 & 0 & 0.010915 & 0 & 0 & 0 & 6.2213e-06 \\
-0.010915 & 0 & 6.2213e-06 & 0 & 0 & 0 & 0 & 0 \\
-6.2213e-06 & 0 & 0.010915 & 0 & 0 & 0 & -6.2213e-06 & 0 \\
0 & 1.7823e-06 & 0 & -0.003127 & 0 & 0 & 0 & -1.7823e-06 \\
0 & 1.0158e-09 & 0 & -1.7823e-06 & 0 & 0 & 0 & 0.003127 \\
-1.7823e-06 & 0 & 0.003127 & 0 & 0 & 0 & -1.7823e-06 & 0 \\
-1.0158e-09 & 0 & 1.7823e-06 & 0 & 0 & 0 & 0.003127 & 0 \\
\multicolumn{8}{c}{O_{3\times 8}}
\end{array}\right] \tag{C.6}
$$

$$D_{11} = \begin{bmatrix} 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0 & 0 & 0 & 0.010143 & 23.598 & 0\,0\,0\,0\,0\,0 & 0 \\ -3.5355 & 0 & -3.5355 & 0\,0\,0\,0\,0\,0\,0 & 0 & 0 & 0 & 0 & 0\,0\,0\,0\,0\,0 & -1.149 \\ 0 & 0 & 0 & 0\,0\,0\,0\,0\,0\,0 & 0.010143 & 23.598 & 0 & 0 & 0\,0\,0\,0\,0\,0 & 0 \\ -3.5355 & 0 & -3.5355 & 0\,0\,0\,0\,0\,0\,0 & 0 & 0 & 0 & 0 & 0\,0\,0\,0\,0\,0 & -1.149 \\ & & & & O_{17\times 21} & & & & & & \end{bmatrix} \tag{C.7}$$

$$D_{12} = \begin{bmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & -1.875 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & -1.875 \\ & & O_{14\times 5} & & \\ 0.000 & 0.000 & 3.491 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 3.491 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.414 \end{bmatrix} \tag{C.8}$$

$$C_2 = \begin{bmatrix} & 0 & -0.003127 & 0 & 0 \\ O_{4\times 4} & -0.003127 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 2093.1 \\ & 0 & 0 & 2093.1 & 0 \end{bmatrix} \tag{C.9}$$

$$D_{21} = \begin{bmatrix} O_{4\times 14} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 286.9 & 6.6745e+05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 286.9 & 6.6745e+05 & 0 & 0 & 0 \end{matrix} \end{bmatrix} \tag{C.10}$$

$$D_{22} = O_{4\times 5} \tag{C.11}$$

# Appendix D

# Synthesized Controllers

## D.1 Continuous Controller $K$

$$K = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{D.1}$$

where $A$, $B$, $C$ and $D$ are state-space matrices.

$$A = \begin{bmatrix}
-170.138 & -0.000 & 743.531 & -0.000 & -26255.506 & 0.000 & 0.000 & -22.745 & 9120.535 & 1.302 & 0.000 & -151.988 \\
0.000 & -170.138 & -0.000 & -743.531 & 0.000 & 26255.506 & 22.745 & 0.000 & -0.017 & 9118.693 & 151.988 & 0.000 \\
-166.606 & 0.000 & -108.483 & 0.000 & 1581.695 & -0.000 & -0.000 & 0.013 & 8246.693 & -0.001 & -0.000 & 0.084 \\
0.000 & 166.606 & -0.000 & -108.483 & 0.000 & 1581.695 & 0.013 & 0.000 & -0.000 & -8246.694 & 0.084 & 0.000 \\
-4.167 & 0.000 & -3.714 & 0.000 & 39.559 & -0.000 & -0.000 & 0.000 & 206.256 & -0.000 & -0.000 & 0.002 \\
0.000 & 4.167 & -0.000 & -3.714 & 0.000 & 39.559 & 0.000 & 0.000 & -0.000 & -206.256 & 0.002 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & -0.000 & 28941.055 & -0.100 & -0.000 & -0.000 & 723.387 & -0.000 & -0.000 \\
0.000 & 0.000 & -0.000 & 0.000 & 28941.055 & -0.000 & 0.000 & -0.100 & -723.387 & 0.000 & -0.000 & -0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 1.000 & -0.000 & -0.000 & -0.000 & -0.025 & 0.000 & 0.000 & -0.000 \\
0.000 & 0.000 & 0.000 & -0.000 & -0.000 & -1.000 & -0.000 & -0.000 & -0.000 & -0.025 & 0.000 & -0.000 \\
0.000 & 0.000 & 0.000 & -0.000 & -0.000 & 0.000 & -0.000 & -0.000 & -0.000 & -0.000 & -0.124 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & -0.000 & -0.000 & -0.000 & -0.000 & -0.000 & 0.000 & -0.000 & -0.124
\end{bmatrix} \tag{D.2}$$

$$B = \begin{bmatrix}
O_{6\times 12} & & & \\
0.000 & 29.150 & -0.000 & 0.000 \\
29.150 & -0.000 & -0.000 & -0.000 \\
0.000 & 0.000 & -0.000 & -0.000 \\
0.000 & 0.000 & -0.000 & 0.000 \\
0.000 & 512.132 & 0.000 & 0.000 \\
512.132 & 0.000 & -0.000 & 0.000
\end{bmatrix} \tag{D.3}$$

$$C = \begin{bmatrix}
0.000 & 0.572 & -0.000 & -0.001 & 0.000 & 0.050 & 0.000 & 0.000 & -0.000 & 0.017 & 0.000 & 0.000 \\
0.572 & -0.000 & 0.001 & -0.000 & -0.050 & 0.000 & 0.000 & -0.000 & 0.017 & 0.000 & 0.000 & -0.000
\end{bmatrix} \tag{D.4}$$

$$D = O_{2\times 4}$$

## D.2 Discrete Controller $K_{disc}$

$$K_{disc} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{D.5}$$

where $A$, $B$, $C$ and $D$ are state-space matrices and the sampling time is 55 ms.

$$A = \begin{bmatrix} -0.937 & 0.000 & 1.812 & 0.000 & -58.107 & -0.000 & 0.000 & -0.039 & 97.806 & 0.002 & 0.000 & -0.261 \\ 0.000 & -0.937 & 0.000 & -1.812 & -0.000 & 58.107 & 0.039 & 0\ 000 & -0.000 & 97.803 & 0.261 & 0.000 \\ -0.070 & -0.000 & -3.143 & -0.000 & 89.147 & 0.000 & -0.000 & 0.044 & 1.248 & -0.003 & -0.000 & 0.292 \\ -0.000 & 0.070 & 0.000 & -3.143 & -0.000 & 89.147 & 0.044 & 0.000 & -0.000 & -1.252 & 0.292 & 0.000 \\ 0.000 & 0.000 & -0.045 & 0.000 & 0.777 & -0.000 & 0.000 & -0.000 & -0.003 & 0.000 & 0.000 & -0.001 \\ 0.000 & -0.000 & -0.000 & 0.045 & 0.000 & 0.777 & -0.000 & -0.000 & 0.000 & 0.003 & -0.001 & -0.000 \\ 0.000 & -0\ 139 & -0.000 & -35.430 & 0.000 & 1409.692 & 0.908 & -0.000 & 0.000 & 42.157 & -0.578 & -0.000 \\ 0.139 & 0.000 & -35.430 & 0.000 & 1409.692 & -0.000 & 0.000 & 0.908 & -42.150 & 0.005 & 0.000 & -0.578 \\ 0.000 & 0\ 000 & -0.001 & 0.000 & 0.049 & -0.000 & 0.000 & -0.000 & 0\ 999 & 0.000 & 0.000 & -0.000 \\ -0.000 & 0.000 & 0.000 & 0.001 & -0.000 & -0.049 & 0.000 & 0.000 & -0.000 & 0.999 & 0.000 & 0.000 \\ -0.000 & 0.000 & 0.000 & 0.000 & -0.000 & -0.000 & -0.000 & 0.000 & 0.000 & 0.000 & 0.993 & 0.000 \\ -0.000 & 0.000 & 0.000 & 0.000 & -0.000 & -0.000 & 0.000 & -0.000 & 0.000 & 0.000 & 0.000 & 0.993 \end{bmatrix} \tag{D.6}$$

$$B = \begin{bmatrix} -67.320 & 0.000 & 0.000 & 0.000 \\ 0.000 & 67.320 & -0.000 & 0.000 \\ 75.370 & -0.000 & -0.000 & -0.000 \\ 0.000 & 75.370 & -0.000 & 0.000 \\ -0.188 & 0.000 & 0.000 & 0.000 \\ -0.000 & -0.188 & 0.000 & 0.000 \\ -0.000 & -120.247 & -0.000 & 0.000 \\ -120.247 & 0.000 & -0.000 & -0.000 \\ -0.005 & 0.000 & -0.000 & -0.000 \\ 0.000 & 0.005 & -0.000 & 0.000 \\ 0.000 & 510.385 & 0.000 & 0.000 \\ 510.385 & 0.000 & -0.000 & 0.000 \end{bmatrix} \tag{D.7}$$

$$C = \begin{bmatrix} -0.000 & 0.001 & 0.000 & -0.029 & -0.000 & 0.914 & 0.001 & 0.000 & -0.000 & 1.539 & 0.004 & 0.000 \\ 0.001 & 0.000 & 0.029 & 0.000 & -0.914 & -0.000 & 0.000 & -0.001 & 1.539 & 0.000 & 0.000 & -0.004 \end{bmatrix} \tag{D.8}$$

$$D = \begin{bmatrix} 0.000 & 1.059 & -0.000 & 0.000 \\ -1.059 & 0.000 & 0.000 & 0.000 \end{bmatrix} \tag{D.9}$$

## D.3 Discrete Controller $K_{n_{disc}}$

$$K_{n_{disc}} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{D.10}$$

where $A$, $B$, $C$ and $D$ are state-space matrices and the sampling time is 55 ms.

$$
A - \begin{bmatrix}
-0.580 & -0.000 & 4.662 & -0.000 & -131.605 & 0.000 & 0.000 & -0.094 & 1149.309 & 0.000 & 0.000 & -0.016 & -0.000 & -1.258 \\
0.000 & -0.580 & 0.000 & -4.662 & -0.000 & 131.605 & 0.094 & 0.000 & -0.000 & 1149.309 & 0.016 & 0.000 & -1.258 & 0.000 \\
-0.016 & 0.000 & 1.300 & 0.000 & 29.340 & -0.000 & -0.000 & 0.004 & 64.913 & -0.000 & -0.000 & 0.001 & 0.000 & -0.072 \\
0.000 & 0.016 & 0.000 & -1.300 & -0.000 & 29.340 & 0.004 & 0.000 & -0.000 & -64.913 & 0.001 & 0.000 & 0.072 & 0.000 \\
0.000 & -0.000 & -0.049 & -0.000 & 0.927 & 0.000 & 0.000 & -0.000 & -0.162 & 0.000 & 0.000 & -0.000 & -0.000 & 0.000 \\
-0.000 & -0.000 & 0.000 & -0.049 & 0.000 & 0.927 & -0.000 & -0.000 & 0.000 & 0.162 & -0.000 & -0.000 & -0.000 & -0.000 \\
-0.000 & -0.031 & 0.000 & -39.080 & -0.000 & 1528.175 & 0.987 & -0.000 & 0.000 & 168.276 & -0.001 & -0.000 & -0.143 & -0.000 \\
0.031 & 0.000 & -39.080 & 0.000 & 1528.175 & -0.000 & 0.000 & 0.987 & -168.276 & 0.000 & 0.000 & -0.001 & 0.000 & 0.143 \\
0.000 & -0.000 & -0.001 & -0.000 & 0.053 & 0.000 & 0.000 & -0.000 & 0.994 & 0.000 & 0.000 & -0.000 & -0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.001 & -0.000 & -0.053 & 0.000 & 0.000 & -0.000 & 0.994 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.190 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.190 & 0.000 & 0.000 \\
0.000 & 0.259 & 0.000 & -3.041 & -0.000 & 85.851 & 0.061 & 0.000 & -0.000 & 749.742 & 0.010 & 0.000 & 0.179 & 0.000 \\
0.259 & -0.000 & 3.041 & -0.000 & -85.851 & 0.000 & 0.000 & -0.061 & 749.742 & 0.000 & 0.000 & -0.010 & -0.000 & 0.179
\end{bmatrix}
\tag{D.11}
$$

$$B = \begin{bmatrix} -5.393 & 0.000 & 0.000 & 0.000 \\ 0.000 & 5.393 & 0.000 & 0.000 \\ 0.215 & -0.000 & -0.000 & -0.000 \\ 0.000 & 0.215 & 0.000 & 0.000 \\ -0.001 & 0.000 & 0.000 & 0.000 \\ -0.000 & -0.001 & -0.000 & -0.000 \\ -0.000 & 28.644 & 0.000 & 0.000 \\ 28.644 & 0.000 & -0.000 & -0.000 \\ -0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 304.727 & 0.000 & 0.000 \\ 304.727 & 0.000 & 0.000 & 0.000 \\ 0.000 & 3.518 & 0.000 & 0.000 \\ -3.518 & 0.000 & 0.000 & 0.000 \end{bmatrix} \tag{D.12}$$



$$C = \begin{bmatrix} 0.000 & 0.008 & 0.000 & -0.095 & -0.000 & 2.682 & 0.002 & 0.000 & -0.000 & 23.423 & 0.000 & 0.000 & -0.026 & 0.000 \\ 0.008 & -0.000 & 0.095 & -0.000 & -2.682 & 0.000 & 0.000 & -0.002 & 23.423 & 0.000 & 0.000 & -0.000 & -0.000 & -0.026 \end{bmatrix} \tag{D.13}$$

$$D = \begin{bmatrix} 0.000 & 0.110 & 0.000 & 0.000 \\ -0.110 & 0.000 & 0.000 & 0.000 \end{bmatrix} \tag{D.14}$$

## D.4 Discrete Controller $K_{red_{disc}}$

$$K_{red_{disc}} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{D.15}$$

where $A$, $B$, $C$ and $D$ are state-space matrices and the sampling time is 55 ms.

$$A = \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 & -0.000 & -0.000 \\ 0.000 & 0.000 & -0.057 & 0.000 & -0.771 & -0.119 \\ 0.000 & 0.000 & -0.000 & -0.057 & 0.119 & -0.771 \\ 0.000 & 0.000 & 0.771 & -0.119 & 0.346 & -0.000 \\ 0.000 & 0.000 & 0.119 & 0.771 & 0.000 & 0.346 \end{bmatrix} \tag{D.16}$$

$$B = \begin{bmatrix} -1.031 & -0.274 & 0.000 & 0.000 \\ 0.001 & -0.002 & -0.000 & -0.000 \\ -1.592 & 0.328 & 0.000 & 0.000 \\ 0.328 & 1.592 & 0.000 & 0.000 \\ -1.031 & 0.051 & 0.000 & 0.000 \\ 0.051 & 1.031 & 0.000 & 0.000 \end{bmatrix} \tag{D.17}$$

$$C = \begin{bmatrix} -0.000 & -0.000 & 0.018 & 0.088 & -0.003 & -0.057 \\ 0.000 & -0.000 & 0.088 & -0.018 & -0.057 & 0.003 \end{bmatrix} \tag{D.18}$$

$$D = \begin{bmatrix} 0.000 & 0.114 & 0.000 & 0.000 \\ -0.114 & 0.000 & 0.000 & 0.000 \end{bmatrix} \tag{D.19}$$

# Appendix E

# Matlab Functions

## E.1 S-Function of Nonlinear Model

```
function [sys,x0,str,ts] = scammod(t,x,u,flag)

% Set up all the known data
% A combines Ax and Ay and the optical flow
A = [0 0          0       0       0          0       0       0
     0 0          0       0       0          0       0       0
     0 0      -68.9    -1363   -8291          0       0       0
     0 0          1       0       0          0       0       0
     0 0          0       1       0          0       0       0
     0 0          0       0       0      -68.9    -1363   -8291
     0 0          0       0       0          1       0       0
     0 0          0       0       0          0       1       0];

% output is [x y Theta_x Theta_y]'
C=[1 0 0      0    0       0       0    0
   0 1 0      0    0       0       0    0
   0 0 0 -207.3 8291       0       0    0
   0 0 0      0    0       0 -207.3 8291];

% Input matrix without disturbnace
Bw=[0    0
   0    0
   1    0
   0    0
   0    0
   0    1
   0    0
   0    0];

B= [0    0    0    0    0
    0    0    0    0    0
    1    0    0    0    0
    0    0    0    0    0
    0    0    0    0    0
    0    1    0    0    0
    0    0    0    0    0
    0    0    0    0    0];
```

```
% Camera Parameters: [nx ny]
N=[.005 .005];

% distance to the object
Zs=2;

switch flag,

  %%%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes();

  %%%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,x,u,A,B,C,N,Zs);

  %%%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%%
  case 3,
    sys=mdlOutputs(t,x,u,C);

  %%%%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%%%
  case { 2, 4, 9 },
    sys=[]; % do nothing

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error([['Unhandled flag = ',num2str(flag)]]);

end

% end sfuntmpl

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts]=mdlInitializeSizes()

sizes = simsizes;
sizes.NumContStates  = 8;   % [x y x_1x x_2x x_3x x_1y x_2y x_3y]
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;   % [x y Theta_x Theta_y]  defined in C matrix
sizes.NumInputs      = 5;   % [Theta_x Theta_y Tx Ty Tz]'
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
```

```
sys = simsizes(sizes);

% initialize the initial conditions
x0  = zeros(sizes.NumContStates,1);
str = [];
ts = [0 0];

% end mdlInitializeSizes

%
%=============================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
%
function sys=mdlDerivatives(t,x,u,A,B,C,N,Zs)

%Compute the nonlinear portions of A
A(1,3)=x(1)*x(2)*N(2)*C(3,4)*pi/180;
A(1,4)=x(1)*x(2)*N(2)*C(3,5)*pi/180;
A(1,6)=-(1/N(1)+x(1)^2*N(1))*C(4,7)*pi/180;
A(1,7)=-(1/N(1)+x(1)^2*N(1))*C(4,8)*pi/180;

A(2,3)=(1/N(2)+x(2)^2*N(2))*C(3,4)*pi/180;
A(2,4)=(1/N(2)+x(2)^2*N(2))*C(3,5)*pi/180;
A(2,6)=-x(1)*x(2)*N(1)*C(4,7)*pi/180;
A(2,7)=-x(1)*x(2)*N(1)*C(4,8)*pi/180;

% Compute nonlinear portion of B
B(1,3)=1/(Zs*N(1));
B(1,5)=-x(1)/Zs;
B(2,4)=1/(Zs*N(2));
B(2,5)=-x(2)/Zs;

% Compute the new states
sys = A*x+B*u;

% end mdlUpdate

%
%=============================================================================
% mdlOutputs
% Return the block outputs.
%=============================================================================
%
function sys=mdlOutputs(t,x,u,C)

sys = C*x;
% sys = [1; 1];
% end mdlOutputs
```

## E.2 Second Order Approximation of Step Response Data

```
% The input is organized as follows:
% data - output of the system
% time - time vector
% T - delay in seconds
% N - order of a Pade approximation of the delay
% The output of the function is:
% G - fitt the system
% error - sum of squared errors between the data and output of G

function (G error)=second_order_fit(data, time, T, N)

data = -1*data;
m = min(data);
data = data - m;
% find the final value
m = max(data);
data=1-data/m;
A = 1;
% level the data
d = A-data;
% remove zeros at the end of the sequence
count = sum(d~=0);
d0=d(1:count);
% find the data to fit, start in the middle
stop = count;
start = floor(stop/2);
% fit the log of the data to find first pole
d0_log=log(d0);
fresult=fit(time(start:stop),d0_log(start:stop),'poly1');
% first pole
p1=-fresult.p1
% find B
B_vect=(data-A)./exp(-p1*time);
B=mean(B_vect(start:stop));
% compute C
C=-(A+B);
% second pole
p2=-B/C*p1
% gain
K=A*p1*p2
G1=tf(K,[1 (p1+p2) (p1*p2)]);
[num den)=pade(T,N);
% transfer function
G=G1*tf(num,den);
[y t]=step(G,time);
plot(time,data,'-..r')
hold on
plot(time,y)
err=data-y;
% sum of square errors
error = err'*err;
```